Data Storage Networking

Real-World Skills for the CompTIA Storage+™ Certification and Beyond

Nigel Poulton

Foreword by Hu Yoshida, VP and CTO of Hitachi Data Systems

Exam SG0-001

SYBEX

A Wiley Brand

Data Storage Networking

Real World Skills for the CompTIA Storage+™ Certification and Beyond

Nigel Poulton

SYBEX
A Wiley Brand

Senior Acquisitions Editor: Jeff Kellum
Development Editor: Kelly Talbot
Technical Editors: Howard Marks and Ron Singler
Production Editor: Eric Charbonneau
Copy Editor: Sharon Wilkey
Editorial Manager: Pete Gaughan
Vice President and Executive Group Publisher: Richard Swadley
Associate Publisher: Chris Webb
Media Project Manager 1: Laura Moss-Hollister
Media Associate Producer: Doug Kuhn
Media Quality Assurance: Shawn Patrick
Book Designers: Judy Fung and Bill Gibson
Compositor: Jeff Lytle, Happenstance Type-O-Rama
Proofreader: Nancy Bell
Indexer: Ted Laux
Project Coordinator, Cover: Todd Klemme
Cover Designer: Ryan Sneed

Published simultaneously in Canada

ISBN: 978-1-118-67921-0
ISBN: 978-1-118-76323-0 (ebk.)
ISBN: 978-1-118-91594-3 (ebk.)

Library of Congress Control Number: 2013958299

10 9 8 7 6 5 4 3 2 1

Dear Reader,
Thank you for choosing *Data Storage Networking: Real World Skills for the CompTIA Storage+ Certification and Beyond*. This book is part of a family of premium-quality Sybex books, all of which are written by outstanding authors who combine practical experience with a gift for teaching.

Sybex was founded in 1976. More than 30 years later, we're still committed to producing consistently exceptional books. With each of our titles, we're working hard to set a new standard for the industry. From the paper we print on to the authors we work with, our goal is to bring you the best books available.

I hope you see all that reflected in these pages. I'd be very interested to hear your comments and get your feedback on how we're doing. Feel free to let me know what you think about this or any other Sybex book by sending me an email at `contactus@wiley.com`. If you think you've found a technical error in this book, please visit `http://sybex.custhelp.com`. Customer feedback is critical to our efforts at Sybex.

Best regards,

[illegible]

Chris Webb
Associate Publisher
Sybex, an Imprint of Wiley

I dedicate this book to the most important people in my life—my wife and children. Thank you so much for being part of my life! And thank you to God, for blessing me with, well, everything!

Acknowledgments

So many people to thank and acknowledge, it's hard to know where to begin!

First and foremost, thank you to my wife and children for being so amazingly supportive and patient with me during this mammoth writing project. Right before I signed the contract to write the book, I pointed out to my wife, Jen, that just about every book I've ever read has started with the words "Thank you to my wife and family who have barely seen me for the last 12 months of writing this book." Being the amazing person that she is, she grimaced and told me to go ahead and do it. Thank you, Jen; I honestly couldn't have done it without your support. Thanks also to my wonderful three daughters, Lily, Abi, and Mia, who have had only half a father for the past 10 months. Looking forward to some time with you over Christmas!

I'd also like to thank my parents for putting up with me as a lazy, delinquent youth. Thank you to my late father for buying me my first PC—a 386SX-33 with half a megabyte of RAM and a 40 MB hard disk, which I was told I'd never be able to fill in my entire life! That computer is what got me hooked on IT. I know you'd be proud of me, Dad. Thank you to my mother, who, as a single parent, went out to work to support her family. Hopefully, I'm paying it all back to you now!

Aside from family, I've had some top-notch help with this book. Thank you to Howard Marks and Ron Singler, the book's technical editors. Howard, you're a legend, and I look up to you as a true "gray beard." Thanks for keeping me straight throughout the project. Ron, I really appreciate your assistance on this project as well, and thanks again for getting me started blogging all those years ago!

Thank you also to Kelly Talbot, my development editor. I really appreciate the work you've put into this project. Your influence now has me spelling words the American way instead of the British way—scary!

A final thanks to Sybex, and Jeff Kellum, for allowing me to write my first Sybex book. I've had a love of IT books since I learned Windows NT 4 back in the late 1990s. In fact, my wife, Jen, occasionally reminds me that I took Mark Minasi's 1,500-page *Mastering Windows 2000 Server* on our honeymoon with us and proceeded to take it to the pool, the beach, and anywhere else we went! It's nice to be able to give something back and hopefully influence somebody's career in a similar way to the way my career was influenced by Mark's book.

About the Author

Nigel Poulton is a well-known and popular figure in the storage industry. He is the author of a long-running storage blog and hosts a popular technical podcast. Nigel is probably best known for his ambition to tackle deep technical concepts and share them via his website and podcast. Nigel has over 15 years of experience working with IT in a wide variety of industries, including financial services, retail, and government. Nigel has hands-on experience with most things in storage, all the way from pouring cups of tea, resetting passwords, and changing backup tapes, to designing petabyte-scale solutions and having responsibility for live production environments. Nigel lives with his wife and three daughters in England and is available as a consultant.

Nigel can be contacted by email at `nigelpoulton@hotmail.com`, as well as through contact forms or leaving comments on his blog site (`nigelpoulton.com`) and his podcast website (`www.technicaldeepdive.com`).

Contents at a Glance

Contents

Table of Exercises

Foreword

My experience with storage began in the early 1960s when I was a lab technician at the University of California. I spent late nights operating an IBM 704 computer, processing data from Bevatron bubble chamber experiments and storing it onto IBM 727 tape drives. Registers were manually set through an octal keyboard; programs and data were loaded through a punched card reader; and I was the operating system, loading card decks, mounting and dialing in the tape drives, and servicing the output printers. The 1960s were a period of social and political unrest on the Berkeley campus, and I ended up leaving the university to join the military service, far removed from storage technology.

After five years, I left the military to seek a career in the computer industry. I applied at IBM, hoping to leverage the computer experience from my university days. During my first interview, I was asked what I knew about DASD. I was stunned. In a relatively short span of five years, the world I knew of punched cards, tape storage devices, and printed output was replaced by something called a direct-access storage device. This was my introduction to disk drives. I felt like Rip Van Winkle, who awoke from a 20-year sleep to find the world completely changed. Ever since then, I often wondered if this type of transformation would occur again, and I would wake up one day to find an entirely new world of storage.

In the 40 some years since my introduction to disk storage, I worked for IBM and then for Hitachi Data Systems in the storage industry and witnessed the transformation of storage to where it is today. This transformation did not occur overnight. It happened year after year, and the transformation continues to move forward. While the primary data storage device continues to be a mechanical device that records data as magnetic bits on a rotating disk platter, a tremendous amount of technology has been developed to increase recording densities from 2,000 bits per in^2 to 10^{12} bits per in^2 (2 Kb/in^2 to 1 Tb/in^2). In the last few years, solid-state disks based on flash technology were introduced for high-performance storage requirements. While SSDs were initially 10 times the cost per bit of high-performance hard disks, the price gap is closing as the cost of SSDs is declining faster than hard disk prices.

IT is going through a major transformation today as the confluence of Big Data, cloud computing, social networks, and mobile devices change the way we work with and process information. These changes are driving an accelerating need for more storage. This need for storage goes beyond performance and capacity. It requires technologies to make the storage of data more efficient, more reliable, more available, and simpler to manage, procure, protect, secure, search, replicate, and access as blocks, files, or objects. Storage is much more than the media that holds the data bits.

This book helps the reader understand these requirements for the storage of data today, including the latest storage technologies and how they work together. This is a reference manual for IT professionals who need to make informed decisions in the investment of storage choices.

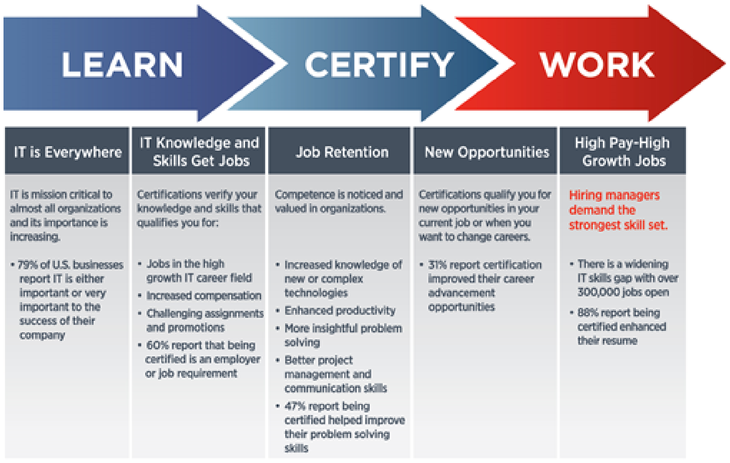
Hu Yoshida
Vice President and Chief Technical Officer
Hitachi Data Systems

CompTIA

It Pays to Get Certified

In a digital world, digital literacy is an essential survival skill.

Certification proves you have the knowledge and skill to solve business problems in virtually any business environment. Certifications are highly valued credentials that qualify you for jobs, increased compensation, and promotion.



IT is Everywhere	IT Knowledge and Skills Get Jobs	Job Retention	New Opportunities	High Pay-High Growth Jobs
IT is mission critical to almost all organizations and its importance is increasing.	Certifications verify your knowledge and skills that qualifies you for:	Competence is noticed and valued in organizations.	Certifications qualify you for new opportunities in your current job or when you want to change careers.	Hiring managers demand the strongest skill set.
• 79% of U.S. businesses report IT is either important or very important to the success of their company	• Jobs in the high growth IT career field • Increased compensation • Challenging assignments and promotions • 60% report that being certified is an employer or job requirement	• Increased knowledge of new or complex technologies • Enhanced productivity • More insightful problem solving • Better project management and communication skills • 47% report being certified helped improve their problem solving skills	• 31% report certification improved their career advancement opportunities	• There is a widening IT skills gap with over 300,000 jobs open • 88% report being certified enhanced their resume

CompTIA Storage+™ POWERED BY SNIA™

- **The CompTIA Storage+ Powered by SNIA certification** exam covers the knowledge and skills required to configure basic networks to include archive, backup, and restoration technologies.
- **Additionally, CompTIA Storage+ Certified candidates** will be able to understand the fundamentals of business continuity, application workload, system integration, and storage/system administration, while performing basic troubleshooting on connectivity issues and referencing documentation.

Steps to Getting Certified and Staying Certified

Review Exam Objectives	Review the certification objectives to make sure you know what is covered in the exam. `http://certification.comptia.org/Training/testingcenters/examobjectives.aspx`
Practice for the Exam	After you have studied for the certification, take a free assessment and sample test to get an idea of what type of questions might be on the exam. `http://certification.comptia.org/Training/testingcenters/samplequestions.aspx`
Purchase an Exam Voucher	Purchase your exam voucher on the CompTIA Marketplace, which is located at: `http://www.comptiastore.com/`
Take the Test	Select a certification exam provider and schedule a time to take your exam. You can find exam providers at the following link: `http://certification.comptia.org/Training/testingcenters.aspx`
Stay Certified! Continuing Education	The CompTIA Storage+ certification is valid for three years from the date of certification. There are a number of ways the certification can be renewed. For more information go to: `http://certification.comptia.org/getCertified/stayCertified.aspx`

How to Obtain More Information

Visit CompTIA online `www.comptia.org` to learn more about getting CompTIA certified.

Contact CompTIA Call 866-835-8020 ext. 5 or email `questions@comptia.org`.

Join the IT Pro Community `http://itpro.comptia.org` to join the IT community to get relevant career information.

Connect with us: We're on LinkedIn, Facebook, Twitter, Flickr, and YouTube.

Introduction

As one of the three pillars of IT infrastructure (computing, networking, and storage), storage is fundamental to just about every IT infrastructure ever built. This makes managing storage an essential skill for everyone involved in IT infrastructure, from systems administrators to infrastructure architects.

This book was written so that you can gain the required storage knowledge from which to build a successful IT career. Understanding how disk drives, flash memory, and storage arrays work is vital to understanding how to design, implement, administer, and troubleshoot storage environments. The book also includes many real-world examples to help clarify concepts and show how they should be applied. Finally, the book covers all the objectives listed in the CompTIA Storage+ exam (SGO-001). So after reading this book, you will have all the knowledge required to both pass the exam and implement storage in the real world.

We dedicate entire chapters to topics such as cloud and converged infrastructure, which are reshaping the IT world. Software-defined storage is also addressed in various chapters.

This book is written with the assumption that you have very little knowledge of storage and want to get up to speed as quickly as possible. Great effort was made so that fundamentals would not be tedious for any reader who feels they already have a good grasp of the basics but want to deepen their knowledge. Covering the basics in a succinct manner also allows such readers to quickly refresh their knowledge and potentially clarify any areas where they were previously uncertain.

Who Should Buy This Book

Anyone with a career or aspiring to have a career in IT infrastructure should buy this book. That career could specialize in data storage, specialize in operating systems, or include responsibilities that cover all three of the major IT infrastructure components (computing, networking, storage). The reason this book and its topics are important to everyone involved with IT infrastructure is that storage is vital, and you cannot be a true IT infrastructure specialist without a solid understanding of storage. Gone are the days when storage design and management could be overlooked or glossed over.

For the reader who is specializing in storage, this book will cover all the knowledge you require in order to slingshot you a long way down the path to becoming a storage expert. You will want to read all chapters of the book and ensure you are proficient in the material covered in each chapter. Fortunately, each chapter is written so that it can be read on its own (without having to read all preceding chapters).

For the reader who wishes to have a job covering all aspects of IT infrastructure (the IT infrastructure generalist), you may want to use this book as a reference manual and dip in and out of the chapters when necessary. For example, if you need to know how to deploy an iSCSI SAN, read the chapter on iSCSI SAN. Or if you need to deploy a new backup solution, read the chapter on backup and recovery. As mentioned previously, each chapter is written so that you can read it independently.

Finally, anybody who wishes to achieve the CompTIA Storage+ certification will find that this book covers all the exam's required topics, plus more.

How This Book Is Organized

This book consists of 15 chapters, plus supplementary information and materials: an extensive glossary, an online bonus exam, and 100 online flash cards to help you remember the information covered in the book and prepare you for the CompTIA Storage+ exam. The chapters are organized as follows:

- Chapter 1, "Storage Primer," stresses the important role of storage in today's world, where high-performance, always-on technology infrastructure is vital to the operation and success of just about every organization.
- Chapter 2, "Storage Devices," provides in-depth coverage of the inner workings of the two major storage mediums in use today: the mechanical disk drive and solid-state media.
- Chapter 3, "Storage Arrays," shifts the focus from individual drives to arrays with massive numbers of pooled drives, and shows how they can improve performance, reliability, and efficiency.
- Chapter 4, "RAID: Keeping Your Data Safe," talks about how RAID technology has kept the world's data safe for more than a quarter of a century, discusses the various options that have been around for most of those 35+ years, and discusses options for the future.
- Chapter 5, "Fibre Channel SAN," discusses the pros and cons of the Fibre Channel SAN, including deep knowledge of the FC protocol as well as how to deploy it in your organization.
- Chapter 6, "iSCSI SAN," discusses the pros and cons of iSCSI SAN technologies and the various options available when deploying iSCSI, including the all-important network aspect.
- Chapter 7, "Files, NAS, and Objects," covers the two major file-sharing protocols, SMB and NFS, and how they are deployed and secured, as well as introducing the concepts of object storage, compliance archives, and content-addressable storage.
- Chapter 8, "Replication Technologies," covers both synchronous and asynchronous replication technologies implemented at various layers of the infrastructure stack (array, host, hypervisor, application) and the pros and cons of each.

- Chapter 9, "Storage Virtualization," discusses how virtualization is implemented at various layers of the storage stack, with emphasis on controller-based virtualization, and a discussion of software-defined storage (SDS).
- Chapter 10, "Capacity Optimization Technologies," talks about the impact of compression, deduplication, thin provisioning, and tiering technologies, including how they affect performance and work with solid-state technologies.
- Chapter 11, "Backup and Recovery," covers the vital twin topics of backup and recovery and discusses how cloud service can be exploited.
- Chapter 12, "Storage Management," focuses on capacity management, performance management, and general management protocols as they relate to storage.
- Chapter 13, "The Wider Data Center Neighborhood," outlines the data center technologies that most closely relate to storage. Topics such as airflow, cabling, rack stability, and more are discussed.
- Chapter 14, "Converged Networking," covers the emerging area of Fibre Channel over Ethernet and the benefits of a single network for traditional LAN and SAN traffic.
- Chapter 15, "Cloud Storage," deals with the increasingly important topic of cloud services. Public cloud, private cloud, and hybrid cloud are all covered.
- The glossary provides concise definitions of all the important terms relating to storage and is a great place for a quick refresher if you can't remember what a term or acronym refers to. This may also be a great resource to read the morning of your exam, should you choose to take the CompTIA Storage+ exam.

Bonus Contents

This book has a web page that provides several additional elements. Items available among these companion files include the following:

Bonus Exam This book comes with a 100-question bonus exam to help you test your knowledge and review important information.

Electronic Flash Cards The digital companion files include 100 questions in flash-card format (a question followed by a single correct answer). You can use these to review your knowledge.

Glossary The key terms from this book, and their definitions, are available as a fully searchable PDF.

NOTE You can download all these resources from `www.sybex.com/go/datastoragenetworking`.

Conventions Used in This Book

This book uses certain typographic styles and other elements to help you quickly identify important information and to avoid confusion. In particular, look for the following style:

- *Italicized text* indicates key terms that are described at length for the first time in a chapter. (Italics are also used for emphasis.)

In addition to this text convention, a few conventions highlight segments of text:

NOTE A note indicates information that's useful or interesting but that's somewhat peripheral to the main text.

TIP A tip provides information that can save you time or frustration and that may not be entirely obvious. A tip might describe how to get around a limitation or how to perform an unusual task.

WARNING Warnings describe potential pitfalls or dangers. If you fail to heed a warning, you may end up spending a lot of time recovering from a problem.

Sidebars

A sidebar is like a note but longer. The information in a sidebar is useful, but it doesn't fit into the main flow of the text.

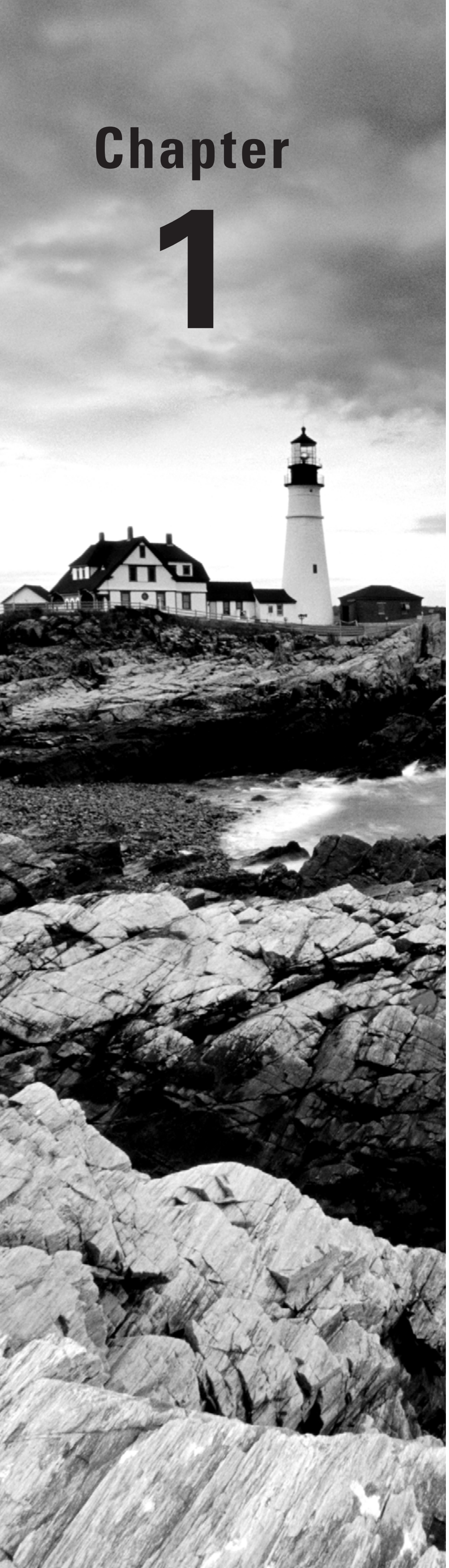
Real World Scenario

Real-World Scenario

A real-world scenario is a type of sidebar that describes a task or example that's particularly grounded in the real world. This may be a situation I or somebody I know has encountered, or it may be advice on how to work around problems that are common in real, working environments.

EXERCISES

An exercise is a procedure you should try out on your own to help you learn about the material in the chapter. Don't limit yourself to the procedures described in the exercises, though! Try other procedures to really learn about data storage networking.



Chapter 1

Storage Primer

TOPICS COVERED IN THIS CHAPTER:

- ✓ The importance of IT
- ✓ The role of storage
- ✓ Types of storage
- ✓ Where storage fits into the wider IT infrastructure
- ✓ The influence of flash memory
- ✓ Cloud storage
- ✓ CompTIA Storage+ Powered by SNIA exam

This introductory chapter outlines the increasingly important role of technology in the world and how this creates a demand for robust, high-performance, technology-based systems that need good, knowledgeable people to design and administer them. It explains the important place that storage occupies in the world of information technology infrastructure and how it interacts with other infrastructure components such as computing and networking. You will learn about the major forms of storage, including the mechanical disk drive and the new wave of solid-state storage technologies such as flash memory. You will also learn about the influence of the cloud.

The Importance of Information Technology

Information technology (IT) has never been more important than it is now. It is more important than it was yesterday and shows all the signs of being even more important tomorrow. From social media and your personal life, all the way up to major governments and corporations, information and technology are at the center of everything.

In our personal lives, we're increasingly reliant on our technology devices—smartphones, smart watches, tablets, laptops, and more. Think about how inconvenienced you were the last time your home Internet connection was down. Suddenly you were cut off from the world, isolated, severely limited in what you could do! Even when we're on the move, we often rely on services such as Google Maps when navigating around big cities or driving our cars. If we're running late from a meeting and need to know the time of the next train home, where do we turn? Our technology devices, of course. It's a fact that we're increasingly reliant on technology in our personal lives.

The same goes for businesses. Businesses are increasingly dependent on information and technology. Try cutting most businesses off from the Internet and see what happens to them! It's not just the Internet. Cut a business off—in any shape or form—from its information and data, and see how long it takes until that business goes bust!

It was not that long ago that high technology was a nice-to-have feature for most businesses. High-speed Internet connections were a luxury, and instant access to information was something that was nice to have. In today's world, businesses simply can't exist without instant access to information and the Internet. In today's world, technology *is* the business. No technology, no business!

As a result of this paramount importance and central nature of technology to businesses, it's vital that businesses have reliable, high-performance technology systems. A low-latency trading company needs super-fast IT systems that enable them to execute as quickly as possible on trades; if they trade too slowly, they stand to lose money. Service companies need reliable, high-performance IT systems that enable them to provide their services to customers; if those services are unavailable or slow, customers will go elsewhere. Online sales companies need high-performance, reliable IT systems to process customer orders; if the online purchasing system is down or too slow, customers won't be able to buy goods. Hospitals need high-speed, reliable access to patient information in order to make accurate clinical decisions; terrible things can happen to patients if doctors can't access up-to-date, accurate patient data in a timely manner. Transportation companies need high-speed, dependable access to IT systems in order to provide safe and reliable services such as air-traffic control and train-track signaling. Media companies need high-speed, reliable IT systems to create and deliver high-quality content to subscribers. The list could go on and on.

IT is important, and that fact is not about to change any time soon. In order for companies to have reliable, high-speed IT systems, they need good people to design and manage them, and that's where you come into the picture!

The Role of Storage in IT

Within the world of IT, *storage* is one of the three major IT infrastructure systems:

- Computing
- Networking
- Storage

At a high level, we often think of these as three layers: the computing layer, the network layer, and the storage layer. In the computing layer, applications such as web servers, databases, and apps live and run. The network layer provides the connectivity between computing nodes. For example, a common design approach has a computer running a web server service, which talks to a database service running on another computer, all via the network layer. Finally, the storage layer is where all the data resides. A simple analogy is as follows: computing nodes are like vehicles such as cars and trucks; the network layer is like the road infrastructure; and the storage layer is like the offices and warehouses that hold all the goods and stock. This analogy might be a bit crude, but it conveys the basic idea.

Persistent and Nonpersistent Storage

There are several types of storage, but at a high level there are two main types:

- Persistent (sometimes called nonvolatile)
- Nonpersistent (sometimes called volatile)

As the names suggest, *persistent storage* is the standard choice for long-term storage of data. We call it *persistent,* or *nonvolatile,* because it doesn't lose its contents when the power is turned off. The spinning disk drive is still probably the most popular form of persistent storage, although flash memory is becoming increasingly popular. Just about every modern personal technology device—smartphone, smart watch, tablet, laptop, and so on—has internal flash memory. Because it's nonvolatile, when the battery runs out on that device (which happens all too often), your data is safe!

Probably the most common example of *nonpersistent,* or *volatile,* storage is random access memory (RAM). Just about all forms and variations of RAM lose their content when the power is removed. So why bother with RAM? Well, it's fast—quite often a lot faster than many forms of persistent storage. At certain types of operation, RAM can be a million times faster than disk and a thousand times faster than flash.

Generally speaking, when we refer to *storage*, we are referring to persistent, nonvolatile storage, which is what the majority of this book focuses on. Occasionally you may hear somebody refer to RAM as storage, but it's rare. Most of the time *storage* refers to persistent storage, whereas *memory* refers to nonpersistent technology such as RAM. An exception to this is flash memory, which is a form of persistent memory meaning that we consider it storage (it doesn't lose its contents when the power goes out) but behaves a lot like memory and is what we call *solid state*.

Although large chunks of this book are devoted to solid-state storage (SSS), we almost always use the term *solid state* to indicate that a device is semiconductor-based and has no moving parts. On the other hand, the disk drive is not a solid-state device because it has many complex moving parts and is therefore considered *mechanical*.

Performance and Availability

Both mechanical and solid-state storage devices have two important characteristics:

- Performance
- Availability

On the performance front, like everything else in IT, it's important that storage is fast. But performance can be a somewhat complicated subject in the storage world. For example, the disk drive is that last remaining mechanical component in the computer, and as such, will always struggle to keep pace with its silicon-based neighbors (CPU, memory, and so on). Because of all the mechanical components involved in every spinning disk drive, the disk drive just doesn't perform well at random workloads. However, it performs excellently at sequential workloads. Flash memory, on the other hand, is like greased lightning when it comes to random-read workloads, and the latest drives based on flash memory are getting faster and faster at sequential workloads, too (some rated at over 400 MBps vs. the latest 15K spinning drives rated at closer to 250 MBps). So storage performance isn't a simple matter and requires a good level of knowledge of the underlying components. Of course, this book addresses all of this, so if these topics seem complex and daunting right now, they should be as clear as crystal after you've read the relevant sections of the book.

On the topic of availability, because of the central and crucial nature of storage in the IT world, it's of paramount importance that storage solutions are designed to deal with component failures such as failed drives. It's a fact of life that things break, and as amazing a feat of modern engineering as the spinning disk drive is, spinning drives still break. And when they break, they tend to break in style, usually taking all their data with them. So it's vital that you understand the concepts that will allow you build highly available, resilient storage solutions. Again, this topic is covered throughout the book.

50-Something Years Young

The disk drive has been around for a long time. There's a good chance that it's older than you. Although it's advanced a lot over the years, it also hasn't changed too much.

Does it sound like the previous sentence contradicts itself? Let me explain. In 1956, IBM manufactured and shipped the IBM RAMAC 350 disk storage unit. This was essentially the first-ever disk drive and the father of all mechanical disk drives. By today's standards, it was a bulky beast. Weighing in at over a ton, it stood 5 feet 6 inches tall and sported no less than 24 50-inch platters (disks). Despite its size and weight, it stored less than 4 MB of data (quite a lot at the time). In terms of capacity and physical dimensions, things have advanced by leaps and bounds since then. But architecturally speaking, the physical design of platters, heads, actuators, and spindle motors has stayed pretty much the same.

During the past 50+ years, the mechanical disk drive has entrenched itself as the de facto medium of choice for high-performance, high-capacity persistent storage requirements. Up until around 2009/2010, it was relatively unchallenged in this category. However, significant changes are ripping their way through the storage world, predominantly because of the following two technologies:

- Solid-state storage/flash memory
- The cloud

Solid-state storage is challenging the supremacy of the spinning disk drive. It is muscling its way into not only personal electronics such as phones and tablets, where it has eradicated the use of the mechanical disk drive, but also the corporate data center. It's not uncommon for laptops and PCs to ship with solid-state drives (SSDs) rather than disk drives, and many storage arrays in the data center are either all-flash (no disk drives) or a combination of spinning disk and SSD.

The cloud is both interesting and extremely disruptive, albeit more disruptive to corporate and personal IT in general, rather than the disk drive per se. However, whereas a user or organization would once naturally have had data stored on its own spinning disk in its own data centers and equipment, this data is increasingly being stored in the cloud, where technology choices such as disk technologies are somebody else's responsibility.

Although the venerable disk drive is over half a century old and still going strong, its dominance in the data center and IT devices is being threatened more than ever before. These are exciting times!

The Influence of the Cloud

The cloud is here and cannot be ignored. In fact, the cloud is exciting and is changing the way we design, deploy, and interact with IT. These aren't just empty promises. The cloud is already here and is delivering on many of its promises.

In a traditional IT infrastructure, a company's IT assets—their servers, storage, and network equipment—were owned and managed by the company's IT department. The equipment was typically housed in a computer room or data center and accessed remotely via network links leased by the company. A typical example is a sales company with an office in New York City that is connected to its servers and applications hosted in the company's own data center somewhere else, such as New Jersey, Boston, or Chicago. In a cloud model, things are different. The company may still have its office in New York, but instead of owning and managing all of its own IT equipment in its own data centers, that equipment is now owned and managed by a cloud provider and sits in the cloud provider's data centers. That cloud provider could be a public cloud company such as Amazon or Rackspace, a private cloud provider, or a software-as-a-service provider such as Salesforce.com or Google Apps.

In the cloud model, it's the responsibility of the cloud provider to purchase IT equipment, manage it on a day-to-day basis, upgrade it, deal with failures, manage the data center, and perform all the traditional IT functions. This allows the sales company to get on with its core line of business and not have to worry about managing IT equipment. Instead of the sales company having to buy equipment up front to deal with anticipated growth and demand, it can pay for the use of cloud-based IT resources on a month-by-month basis. For example, imagine that the sales company has moved its IT infrastructure to the cloud and does a steady trade for most of the year but has huge peaks in sales around Black Friday, Cyber Monday, and the Christmas holiday season. If the company's normal level of trade requires 100 virtual servers and 500 TB of storage, but that requirement peaks for the holiday periods to 250 virtual servers and 750 TB of storage, the cloud is a great choice. This is because for most of the year, the company uses and pays for 100 virtual servers and 500 TB of storage, but then for the holiday periods, it increases the number of virtual servers to 250 and the amount of storage to 750 TB. When the holiday periods are over, the company turns off the additional servers and storage and stops paying for them until the same time next year.

This is just one example of how the cloud is changing the way we consume and pay for IT services. This topic is covered in more detail later in the book.

CompTIA Storage+ Powered by SNIA (SGO-001) Exam

This book covers the topics required for the CompTIA Storage+ Powered by SNIA (SGO-001) exam. However, the book is absolutely far more than an exam prep guide. It covers a lot more material than on the exam and is focused on preparing you for working in the real world. That said, I've personally taken, and passed, the exam while writing this book, and I am confident that the book does justice to the materials covered by the exam. But the book also covers more, a lot more.

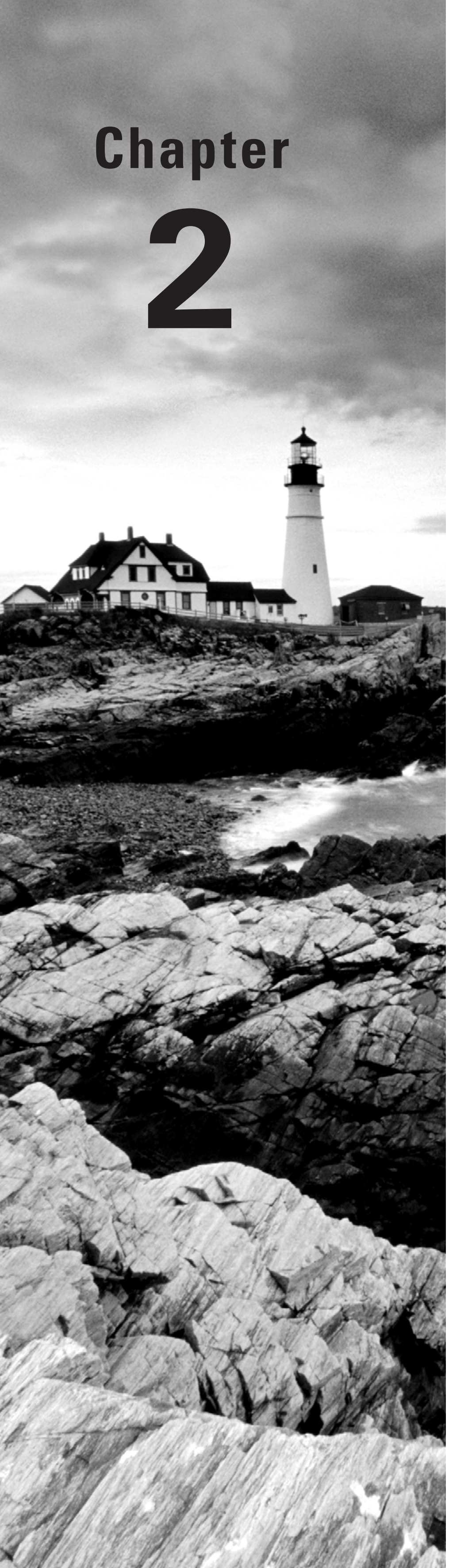
Chapter Essentials

The Importance of Information Technology IT has never been more important in our personal lives and our businesses. As individuals, we increasingly feel isolated and unconnected without our technology devices. Likewise, businesses are increasingly dependent on technology. The concept of no technology = no business has never been more true.

Persistent Storage Persistent storage, or a persistent storage medium, is one that doesn't lose the data stored in it when the power is turned off. The disk drive is a prime example. You can turn the power off to a disk drive and power the drive up again a year later, and the data will still be there. The majority of this book focuses on forms of persistent storage and solutions built around persistent storage devices.

Chapter

2



Storage Devices

TOPICS COVERED IN THIS CHAPTER:

- ✓ **Storage from 40,000 feet**
- ✓ **The mechanical disk drive**
- ✓ **Disk drive performance**
- ✓ **Disk drive reliability**
- ✓ **Solid-state media**
- ✓ **Solid-state performance**
- ✓ **Solid-state reliability**
- ✓ **Tape**
- ✓ **Hybrid Drives**

This chapter covers everything you will need to know about disk drives and solid-state drives for both your career in IT as well as the CompTIA Storage+ exam.

Because a solid understanding of the disk drive and solid-state technology is so fundamental to a successful career in storage, this chapter covers a lot of the underlying theory in depth. It covers all the major mechanical components of the spinning disk drive as well as diving into the theory and inner workings of flash memory and solid-state technologies. It introduces you to the wildly different performance characteristics of spinning media vs. solid-state media. Each has its place, and it's not necessarily as simple as saying that flash memory and solid-state technologies will totally replace the spinning disk drive.

After you've covered the theory of how these critical data center technologies work, you'll learn a little bit about their impact on overall performance and availability—which will be crucial to your IT environment. You'll also learn about capacities and explore cost-related concepts such as dollars per terabyte ($/TB) and dollars per I/O operation ($/I/O operation) and how these cost metrics can be used to make business cases for storage spending.

Storage from 40,000 Feet

The goal of this chapter is to cover everything that you will ever need to know about the storage devices that dominate data centers across the planet. Those devices are as follows:

- Disk storage
- Solid-state storage
- Tape storage

All three are forms of *storage*, or *media*.

Disk storage refers to the electromechanical hard disk drive. But as that is a bit of a mouthful, most people refer to it as a *disk drive*, *hard drive*, or *hard disk drive (HDD)*. I'll interchange the terms to keep you on your toes.

Solid-state media refers to a whole bunch of newer technologies that are fighting for a share of the data center oxygen. Many of these are currently flash memory–based, but other forms of solid-state media exist, and many more are on the horizon. This is a particularly exciting area of storage.

Tape storage refers to magnetic tape. As crazy as it may sound, magnetic tape is still found in IT infrastructures across the globe. And like the disk drive, tape shows no signs of giving up its share of the data center oxygen. I expect magnetic tape to outlive me, and I'm planning on a long and happy life.

TIP

Don’t make the mistake of thinking that you can skip this chapter and still be a storage expert. And don’t make the mistake of thinking that technologies such as the disk drive and tape are old tech and you don’t need to understand them inside and out. Take the time required to get a deep understanding of the information contained in this chapter, and it will serve as gold mine for you in your future career as a storage professional.

I think it’s worth pointing out that in enterprise tech, when people refer to storage, they are almost always referring to disk or solid-state storage. Occasionally people might use the term to refer to optical media or tape, but nobody ever uses it to refer to memory. It’s only the nontech novice who thinks their PC has 1 TB of memory (it really has 1 TB of storage).

Disk, solid-state, and tape storage are all forms of *persistent storage*, or *nonvolatile storage*. Both terms mean the same thing: if you throw the power switch off, your data will still be there when you turn the power back on. Of course, you should never just throw the switch on your computer equipment. You need to be careful to turn off the computer in a sensible way, as any data that is only partially written when the power is pulled will be only partially written when the power is restored—a term known as *crash consistent*. This behavior differs from that of *memory,* such as DRAM; if you pull the power to DRAM, you can kiss goodbye the data it once stored!

WARNING

Misunderstanding the term *crash consistent* can be extremely dangerous. Data that is crash consistent is not guaranteed to be consistent at all! In fact, there is a good chance it might be corrupt. Understanding this is especially important when it comes to application backups. Crash consistent is not even close to being good enough.

Now that those basics are done with, let’s get up close and personal with each of these vital storage technologies.

The Mechanical Disk Drive

Hey, we’re well into the 21st century, right? So why are we still rattling on about the mechanical disk drive? It’s a hunk of rotating rust that belongs back in the 20th century, right?

Wrong!

Yes, the disk drive is ancient. Yes, it’s been around for well over 50 years. And yes, it is being pushed to its limits more than ever before. But like it or lump it, the mechanical disk drive is showing no signs of taking voluntary retirement any time soon.

TIP

While spell checkers might not raise any flags if you spell *disk* as *disc*, doing so will expose you as a bit of a storage novice. Nobody worth their salt in the storage industry spells disk with a *c* at the end. Exceptions include references to optical media such DVDs or Blu-ray Discs.

NOTE The IBM 350 RAMAC disk drive that shipped in 1956 was a beast! It had 50 platters, each one 24 inches in diameter, meaning it would definitely not fit in your phone, tablet, or even your laptop. And you wouldn't want it to, either. It needed a forklift truck to be moved around. For all its size and weight, it sported only a pathetic 4 MB of capacity. Fifty 24-inch platters, a forklift truck to move it, and only 4 MB of capacity! Fast-forward to today, where you can have over 4 TB of capacity in your pocket. 4 TB is over a million times bigger than 4 MB. That's what I call progress!

As the disk drive and its quirky characteristics are absolutely fundamental to a solid understanding of storage, I highly recommend you know the disk drive inside and out. If you don't, it will come back to bite you later. You have been warned!

The Anatomy of a Disk Drive

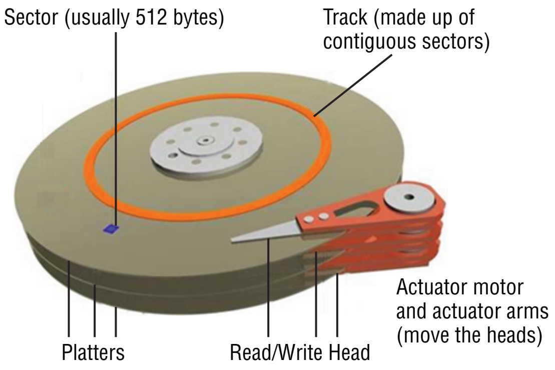
In the vast majority of servers, laptops, and home PCs, the disk drive is the last remaining mechanical component. Everything else is silicon based. This is important to know, because it makes the disk drive hundreds, sometimes even thousands, of times slower than everything else in a server or PC.

The disk drive is composed of four major mechanical components:

- Platters
- Read/write heads
- Actuator assembly
- Spindle motor

Figure 2.1 shows these components. We will also discuss other components in the following sections.

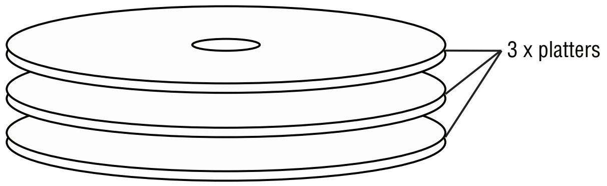
FIGURE 2.1 Basic anatomy of a disk drive


Sector (usually 512 bytes)
Track (made up of contiguous sectors)
Actuator motor and actuator arms (move the heads)
Platters
Read/Write Head

Platter

The *platter* is where the data is stored. A platter looks just like any other rigid, circular disk such as a Blu-ray Disc or DVD. Most disk drives have several platters stacked above each other, as shown in Figure 2.2.

FIGURE 2.2 Three stacked platters



Each platter is rigid (it doesn't bend), thin, circular, insanely flat and smooth, and spins like the wind. Data is read and written to the platter surfaces by the read/write heads, which are controlled by the actuator assembly—more on these later.

Platters are often constructed of a glass or aluminum substrate, and each platter surface is magnetically coated. Both the top and bottom surface of each platter are used to store data. Capacity is so often the name of the game when it comes to disk drives that absolutely no space is wasted, ever!

Because all platters are attached to a common shaft (the spindle), all platters in a disk drive rotate in sync, meaning they all start and stop at the same time and spin at the same speed.

NOTE

Rigidity and smoothness of the platter surface is unbelievably important. Any defect on a platter surface can result in a head crash and data loss. To avoid head crashes, the heads fly at a safe distance above the platter surface while staying close enough to be able to read and write to it. A smooth surface is extremely important here, as it allows the heads to fly closer to the platter surface and get a clearer signal because there is less interference, known as *noise*. A less smooth surface creates more noise and increases the risk of a crash.

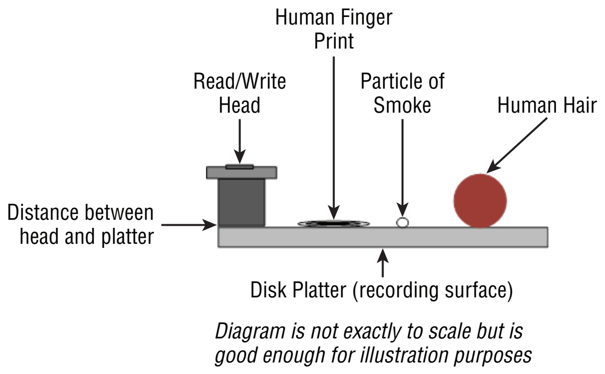
Read/Write Heads

Read/write heads are sometimes referred to as *R/W heads*, or just *heads*. These are the babies that fly above the platter surface and read and write data to the platters. They are attached to the actuator assembly and are controlled by the firmware on the disk drive controller. Higher-level systems such as operating systems, volume managers, and file-systems never know anything about R/W heads.

Flying Height

The read/write heads fly at incredibly minute distances above each platter, referred to as *flying height*. Flying height is measured in nanometers. And if a nanometer doesn't mean anything to you, the flying height of most disk drives is less than the size of a speck of dust or the depth of a fingerprint. Figure 2.3 helps put this in perspective and does a great job of illustrating the precision engineering that goes into the design and manufacturing of the modern disk drive. The disk drive is nothing short of a marvel of modern engineering!

FIGURE 2.3 Head flying height


Human Finger Print
Read/Write Head
Particle of Smoke
Human Hair
Distance between head and platter
Disk Platter (recording surface)
Diagram is not exactly to scale but is good enough for illustration purposes

It is not only the flying height of the heads that is extremely microscopic and precise; the positioning of the heads over the correct sector is also ridiculously precise. The data is so densely packed onto the platter surface these days that being out of position by the most minuscule fraction will see the R/W heads reading or writing to the wrong location on disk. That is called corruption, and you don't want to go there!

Head Crashes

It is vital to note that the read/write heads never touch the surface of a platter. If they do touch, this is known as a *head crash* and almost certainly results in the data on the drive being unreadable and the drive needing replacement.

Reading and Writing to the Platter

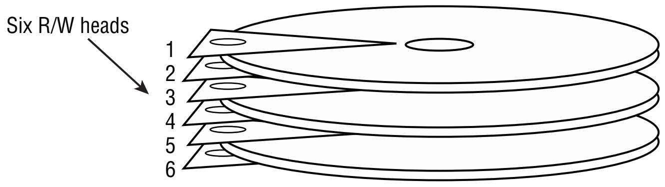
As the heads fly over the surface of the platter, they have the ability to sense and alter the magnetic orientation of the bits in the sectors and tracks that pass beneath them.

When writing data, the read/write heads fly above or below each platter surface and magnetize the surface as they pass over it. Magnetic charging is done in a way to represent binary ones and zeros. When reading data from the platter surfaces, the read/write heads detect the magnetic charge of the areas beneath them to determine whether it represents a binary one or zero. There is effectively no limit to the number of times you can read and write a sector of a disk.

Heads and Internal Drive Addressing

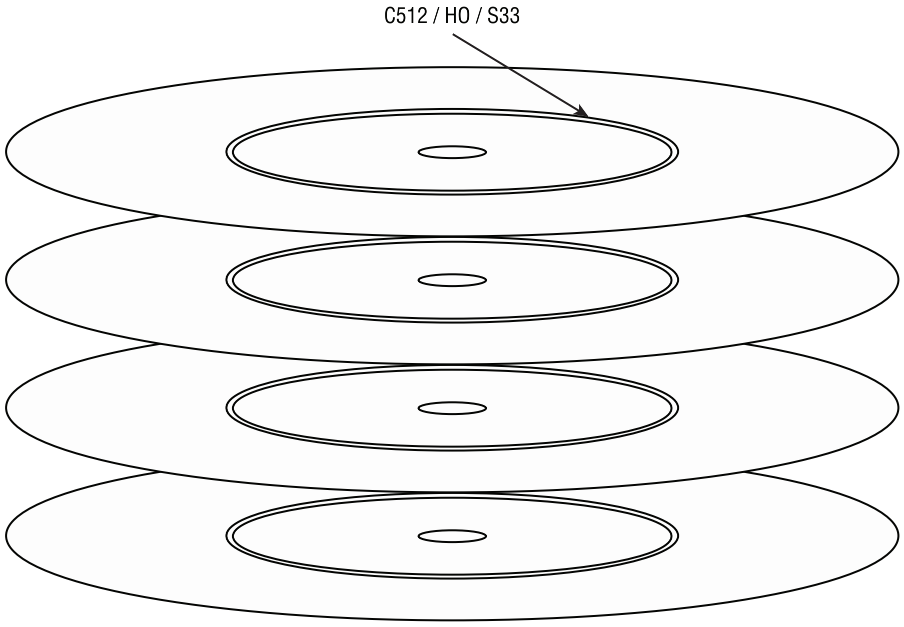
Each platter surface has its own R/W heads. Figure 2.4 shows a drive with three platters and six R/W heads. Each platter has two recording surfaces—top and bottom—and each surface has its own R/W head. Three platters, each with two recording surfaces, equal six recording surfaces. To read and write from these recording surfaces, we need six R/W heads.

FIGURE 2.4 Heads and platter surfaces



This concept of heads and recording surfaces is important in the cylinder-head-sector (CHS) addressing scheme used by disk drives and referenced in the CompTIA Storage+ exam objectives. To address any sector in a disk drive, you can specify the cylinder number (which gives us the track), the head number (which tells us which recording surface this track is on), and the sector number (which tells us which sector on the track we have just identified). Figure 2.5 shows cylinder 512, head 0, sector 33, or put another way, sector 33 on track 512 on platter surface 0.

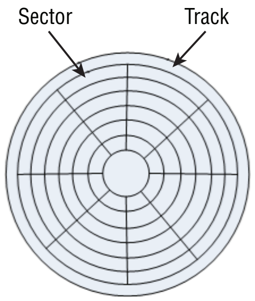
FIGURE 2.5 CHS location



Tracks and Sectors

The surface of every platter is microscopically divided into *tracks* and *sectors*. Figure 2.6 shows an example platter divided into tracks and sectors.

FIGURE 2.6 Aerial view of tracks and sectors



Each platter surface has many tracks, each a concentric ring running all the way around the platter. Each track is then divided into sectors.

The sector is the smallest addressable unit of a disk drive and is typically 512 or 520 bytes in size (we'll work predominantly with 512 bytes in our examples). So, if you want to write only 256 bytes to your disk, you need a whole 512-byte sector and will waste 256 bytes. But don't stress over this; the values are so small as to be insignificant! If you want to write 1,024 bytes, you need four sectors. Simple!

NOTE

While on the topic of sector size, most available Serial Advanced Technology Attachment (SATA) drives have a fixed sector size of 512 bytes, whereas Fibre Channel (FC) and Serial Attached SCSI (SAS) drives can be arbitrarily formatted to different sector sizes. This ability to arbitrarily format different sector sizes can be important when implementing data integrity technologies such as end-to-end data protection (EDP). In EDP, sometimes referred to as *Data Integrity Field* (T10 DIF) or *logical block guarding*, additional data protection metadata is appended to data and stays with that data as it travels from the host filesystem all the way to the storage medium. Disk drives implement this as an additional 8 bytes of data added to the end of every 512-byte sector, making the sector size on drives that support EDP 520 bytes.

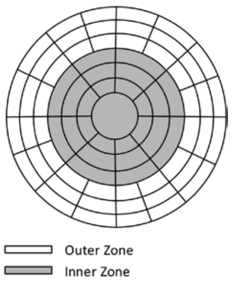
EDP allows drives to detect errors either before committing data to the drive or before returning corrupted data to the host/application. Part of the 8 bytes of data added by EDP is the *guard field*, which includes a cyclic redundancy check (CRC) that allows each device en route from the application to the drive to check the integrity of the data and ensure it hasn't become corrupted.

As you can see in Figure 2.6, the physical size of each sector gets smaller and smaller toward the center of the platter. Because each sector stores the same amount of data (for example, 512 bytes), the outer tracks contain wasted space. Wasting space/capacity in disk drive design is a cardinal sin, so a fix had to be invented. To make better use of available space, most modern disks implement a system known as *zoned data recording (ZDR)*: more sectors are squeezed onto tracks closer to the edges than they are on the inner tracks, allowing the drive to store more data. See Figure 2.7 for a graphical depiction.

In ZDR-style disks—and just about every disk in the world implements some form of ZDR—each platter is divided into *zones*. Tracks in the same zone have the same number of sectors per track. However, tracks in different zones will have a different number of sectors per track.

Figure 2.7 shows a platter surface divided into two zones: the outer zone with 16 sectors per track, and the inner zone with 8 sectors per track. In this simplified example, our platter has three tracks in the outer zone and three tracks in the inner zone, for a total of 72 sectors. If the platter had only a single zone with 8 sectors per track, the same disk would have only 48 sectors, or two-thirds of the potential capacity. Clearly, this is an oversimplified example; in the real world, each track has many more sectors. Even so, this example shows how implementing ZDR techniques can yield greater space efficiencies.

FIGURE 2.7 Zoned data recording



NOTE Modern disk drives can pack more than 1 terabit of data per square inch! A terabit is approximately 128 GB! This is known as *areal density*.

Here are a couple of quick points on performance that are relative to tracks and sectors:

- Because the outer tracks of a platter contain more sectors, it stands to reason they can store and retrieve more data for every spin of the disk. Let's look at a quick example. In our example platter in Figure 2.7, the outer tracks can read or write 16 sectors per rotation, whereas the inner tracks can read or write only 8 sectors per rotation. The outer tracks can perform up to twice as fast as the inner tracks. For this reason, most disk drives will start writing data to the outer tracks of each platter. While this is important theory to know, there is precious little you can do to influence this behavior or exploit your knowledge of it in the real world. My advice is to understand the theory but don't lose any sleep over it!

- If data can be read or written to contiguous sectors on the same or adjacent tracks, you will get better performance than if the data is scattered randomly over the platter. For example, reading sectors 0–15 in the outer track (known as a *sequential read*) could be accomplished in a single rotation of the platter. In contrast, if the 16 sectors being read were scattered all over the platter surface, the read operation (in this case called a *random read*) would almost certainly require more than a single rotation. This can be pretty darn important in real life!

This process of dividing a platter into tracks and sectors is done via a factory format.

NOTE

A *factory format*, sometimes referred to as a *low-level format*, is the process of laying down physical markings on the platter surface. These markings form the actual tracks and sectors that data (ones and zeros) will be written to, as well as control markings, which are used for things like timing and accurate head positioning. This low-level factory format is not the same as the format operation most IT administrators are familiar with. The kind of format frequently performed by system administrators is the process of writing a filesystem to a previously formatted disk drive, which is much more of a high-level operation than a factory format. Most people will never have to perform a low-level format on a disk drive.

Real World Scenario

Short Stroking

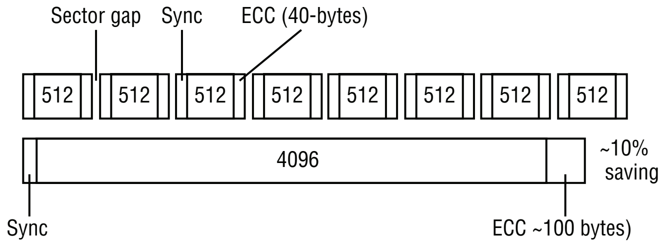
In the real world, people used to *short stroke* disks in order to achieve performance. This process of short stroking involved using only a fraction of the available capacity on the disk, let's say 50 percent. By using only 50 percent of the capacity, you would be using only the outermost racks of the disks (although this is implementation specific, and you had to be sure). You would also be reducing the overall load on the disks, because they had only half the data on them as compared to using 100 percent of the capacity.

However, this practice is rarely used nowadays. First, it is wasteful of capacity, and capacity costs money! Second, in most cases deploying solid-state media is far more effective at increasing performance than short stroking disks.

Each track on a disk has an index marker that allows the read/write heads to maintain accurate positioning—just like lines separating lanes on a highway. Adjacent sectors are separated by *servo signals* that assist in keeping the R/W heads "on track."

While sectors have historically been either 512 bytes or 520 bytes, some more modern disk drives are being formatted with larger sector sizes, such as the increasingly popular Advanced Format standard. Larger sector sizes, currently 4K, offer potentially more capacity as they can implement more-efficient integrity checks such as error correction code (ECC). Figure 2.8 compares a 4K Advanced Format drive with a standard drive formatted with 512-byte sectors.

FIGURE 2.8 4K Advanced Format



In Figure 2.8, the 512-byte format drive requires 320 bytes of ECC, whereas the 4K formatted drive requires only about 100 bytes. Specifics will change and be improved, but this highlights the potential in using larger sectors.

Interestingly, 4K is something of a magic number in x86 computing. Memory pages are usually 4K, and filesystems are often divided into 4K extents or clusters.

The type of disk where the sector size is preset and cannot be changed is referred to in the industry as *fixed block architecture (FBA)*. This term is rarely used these days, though, as these are the only type of disks you can buy in the open systems world.

TIP

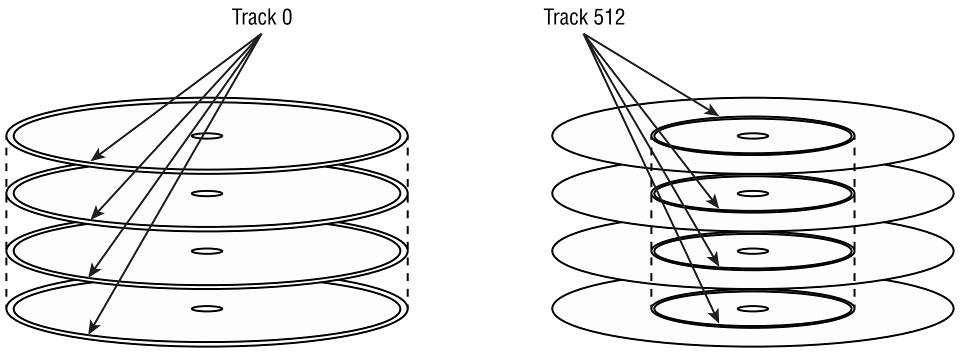
When vendors quote Input/output Operations Per Second (IOPS) figures for their drives, they are often 512-byte I/O operations. This is basically the best I/O operation size for getting impressive numbers on a spec sheet, which is fine, but you may want to remember that this will be way smaller than most real-world I/O operations.

Cylinders

A *cylinder* is a collection of tracks stacked directly above one another on separate platters. As an example, let's assume our disk drive has four platters, each with two recording surfaces, giving us a total of eight recording surfaces. Each recording surface has the same number of tracks, numbered from track 0 on the outside to track 1,023 on the inside (this is just an example for illustration purposes). Track 0 on each of the eight recording surfaces is the outer track. Track 0 on recording surface 0 is directly above track 0 on all seven other recording surfaces.

This concept is shown in Figure 2.9. On the left is a four-platter disk drive with track 0, the outermost track, highlighted. The platters' track 0s form a cylinder that we could call cylinder 0. On the right in Figure 2.9 we have the same four-platter drive, this time with track 512 highlighted, and these tracks combine to form cylinder 512.

FIGURE 2.9 Cylinders



Logical Block Addressing

The way in which data is stored and retrieved from disk can be pretty complex and can vary massively from one disk to another, depending on the characteristics of the disk—sector size, number of platters, number of heads, zones, and so on. To hide these complexities from the likes of operating systems, disk drives implement what is known as *logical block addressing (LBA)*.

Logical block addressing, implemented in the drive controller of every disk drive, exposes the capacity of the disk as a simplified address space—and it is a godsend! Consider our example platter shown previously in Figure 2.7, with two zones and 72 sectors per platter. Let's assume our drive has four platters, giving us a total of eight recording surfaces (eight R/W heads).The locations on our disk drive will be addressed internally in the disk by a CHS address such as these:

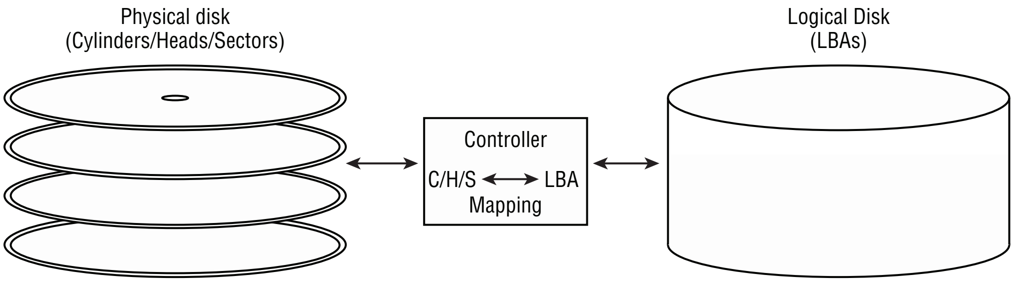
- Cylinder 0, head 0, sector 6
- Cylinder 3, head 6, sector 2
- Cylinder 2, head 0, sector 70

This is ugly, complicated, and specific to each drive type. LBA addressing, on the other hand, couldn't be simpler. The LBA map exposed by the disk controller may simply present a range of 72 logical blocks addressed as LBA0 through LBA71. That's it! Operating systems and filesystems just address reads and writes to LBA addresses without understanding the underlying layout of the disk drive, and the controller on the disk drive takes care of the exact CHS address.

While this might not seem to be very much, consider that all modern disk drives have multiple platters, each with two recording surfaces, and each surface with its own read/write head. You can see how this could quickly become complicated.

Interestingly, because all open-systems operating systems and volume managers work with LBAs, they never know the *precise* location of data on disk! Only the disk controller knows this, as it owns the LBA map. Sound worrying? It shouldn't; the LBA scheme has been used safely and successfully for many years. Figure 2.10 shows how CHS addresses are mapped to LBA addresses on the disk drive controller.

FIGURE 2.10 Disk controller mapping CHS to LBA



A major advantage of the LBA scheme implemented by *intelligent* disk controllers is that operating system drivers for disks have been hugely simplified and standardized. When was the last time you had to install a custom driver for a newly installed disk drive? The OS/driver can simply issue read/write requests, specifying LBA addresses, to the disk, and the controller converts these commands and locations into the more primitive seek, search, read and write commands, as well as CHS locations.

Actuator Assembly

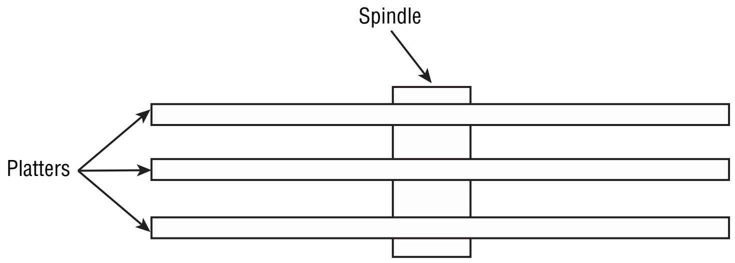
The next important physical component in every disk drive is the *actuator assembly*. Each disk drive has a single actuator assembly, and it is the job of that assembly to physically move the R/W heads (under the direction of the disk drive firmware and controller). And when I say move them, I mean *move them*! If you haven't seen R/W heads flicking back and forth over the surface of a platter, you haven't lived! They move fast and with insane precision. Because a disk drive has only a single actuator, all heads (one for each platter surface) move in unison.

Spindle

Every platter in a disk drive is connected to the *spindle*, and it is the job of the spindle and spindle motor to spin the disks (platters). See Figure 2.11. Each disk drive has only a single spindle, and the speed at which the spindle motor spins the platters is amazingly fast, and again, amazingly precise.

The center of each platter is attached to the spindle, and when the spindle spins, so do the platters. Because there is a single spindle and all platters attach to it, all platters start and stop spinning at the same time—when the spindle starts and stops. Each platter therefore spins at the same velocity, known as revolutions per minute (RPM).

FIGURE 2.11 Cross-section of platters and spindle



Controller

Each disk drive comes with its own controller. This *controller* is like a mini computer just for the disk drive. It has a processor and memory, and it runs firmware that is vital to the operation of the drive.

It is the controller that masks the complexities of the physical workings of the disk and abstracts the layout of data on the drive into the simpler LBA form that is presented up to the higher levels such as the BIOS and OS. Higher levels such as the OS are able to issue simple commands such as read and write to the controller, which then translates them into the more complex operations native to the disk drive. The controller firmware also monitors the health of the drive, reports potential issues, and maintains the bad block map.

NOTE

The *bad block map* is a bunch of hidden sectors that are never exposed to the operating system. These hidden sectors are used by the drive firmware when bad sectors are encountered, such as sectors that cannot be written to. Each drive has a number of hidden blocks used to silently remap bad blocks without higher levels such as operating systems ever knowing.

Interfaces and Protocols

A *protocol* can easily be thought of as a set of commands and rules of communication. Disk drives, solid-state media, and storage arrays speak a protocol. (Disk arrays often speak multiple protocols.) However, in the storage world, each protocol often has its own physical *interface* specifications as well, meaning that protocol can also relate to the physical interface of a drive. For example, a SATA drive speaks the SATA protocol and also has a physical SATA interface. The same goes for SAS and FC; each has its own interface and protocol.

The four most common protocols and interfaces in the disk drive world are as follows:

- Serial Advanced Technology Attachment (SATA)
- Serial Attached SCSI (SAS)
- Nearline SAS (NL-SAS)
- Fibre Channel (FC)

Drive Fragmentation

As a disk drive gets older, it starts to encounter bad blocks that cannot be written to or read from. As these are encountered, the disk drive controller and firmware map these bad blocks to hidden reserved areas on the disk. This is great, and the OS and higher layers never have to know that this is happening. However, this is one of the major reasons that disk drive performance significantly worsens as a computer/disk drive gets older. Let's look at a quick example.

Assume that a single sector on 25 percent of the tracks on a disk is marked as bad and has been remapped to elsewhere on the disk. Any sequential read to any one of those 25 percent of tracks will require the R/W heads to divert off to another location in order to retrieve the data in the one remapped sector. This causes additional and previously unnecessary positional latency (seek time and rotational delay). This is known as *drive fragmentation*, and there is nothing that you can do about it at this level. Running a filesystem defrag in your OS will not help with this scenario. The only thing to do is replace the drive.

All of these are both a protocol and a physical interface specification. Of the latter two, the underlying command set is SCSI, whereas with SATA the underlying command set is ATA. These protocols are implemented on the disk controller and determine the type of physical interface in the drive.

SATA

SATA has its roots in the low end of the computing market such as desktop PCs and laptops. From there, it has muscled its way into the high-end enterprise tech market, but it is quickly being replaced in storage arrays by NL-SAS.

In enterprise tech, SATA drives are synonymous with cheap, low performance, and high capacity. The major reason for this is that the ATA command set is not as rich as the SCSI command set, and as a result not as well suited to high-performance workloads. Consequently, disk drive vendors implement SATA drives with lower-cost components, smaller buffers, and so on.

However, the enterprise tech world is not all about high performance. There is absolutely a place for low-cost, low-performance, high-capacity disk drives. Some examples are backup and archiving appliances, as well as occupying the lower tiers of auto-tiering solutions in storage arrays. However, most of these requirements are now satisfied by NL-SAS drives rather than SATA.

SAS

SAS is, as the name suggests, a serial point-to-point protocol that uses the SCSI command set and the SCSI advanced queuing mechanism. SCSI has its roots firmly in the high-end enterprise tech world, leading to it have a richer command set, better queuing system, and often better physical component quality than SATA drives. All this tends to make SCSI-based drives, such as SAS and FC, the best choice for high-performance, mission-critical workloads.

Of course, performance comes at a cost. SAS drives are more expensive than SATA drives of similar capacity.

Another key advantage of SAS is that SATA II, and newer, drives can connect to a SAS network or backplane and live side by side with SAS drives. This makes SAS a flexible option when building storage arrays.

SAS drives are also dual ported, making them ideal for external storage arrays and adding higher resilience. On the topic of external storage arrays and SAS drives, each port on the SAS drive can be attached to different controllers in the external storage array. This means that if a single port, connection to the port, or even the controller fails, the drive is still accessible over the surviving port. This failover from the failed port to the surviving port can be fast and transparent to the user and applications. Although SAS drives are dual ported, these ports work in an active/passive mode, meaning that only one port is active and issuing commands to the drive at any one point in time.

Finally, SAS drives can be formatted with arbitrary sector sizes, allowing them to easily implement EDP (sometimes called DIF). With EDP, 520-bye sectors are used instead of 512-bye sectors. The additional 8 bytes per sector are used to store metadata that can ensure the integrity of the data, making sure that it hasn't become corrupted.

FC

FC drives implement the SCSI command set and queuing just like SAS, but have dual FC-AL or dual FC point-to-point interfaces. They too are suited to high-performance requirements and come at a relatively high cost per gigabyte, but are very much being superseded by SAS drives.

NL-SAS

Nearline SAS (NL-SAS) drives are a hybrid of SAS and SATA drives. They have a SAS interface and speak the SAS protocol, but also have the platters and RPM of a SATA drive.

What's the point? Well, they slot easily onto a SAS backplane or connector and provide the benefits of the SCSI command set and advanced queuing, while at the same time offering the large capacities common to SATA. All in all, it is a nice blend of SAS and SATA qualities.

NL-SAS has all but seen off SATA in the enterprise storage world, with all the major array vendors supporting NL-SAS in their arrays.

Queuing

All disk drives implement a technique known as *queuing*. Queuing has a positive impact on performance.

Queuing allows the drive to reorder I/O operations so that the read and write commands are executed in an order optimized for the layout of the disk. This usually results in I/O operations being reordered so that the read and write commands can be as sequential as possible so as to inflict the least head movement and rotational latency as possible.

The ATA command set implements Native Command Queuing (NCQ). This improves the performance of the SATA drive, especially in concurrent IOPS, but is not as powerful as its cousin Command Tag Queuing in the SCSI world.

Drive Size

In the storage world, size definitely matters! But it also matters that you know what you're talking about when it comes to disk size.

When speaking of disk drives, there are two types of size you need to be aware of:

- Capacity
- Physical form factor

Normally, when referring to the size of a drive, you are referring to its *capacity*, or how much data it can store. Back in the day, this was expressed in megabytes (MB) but these days we tend to talk in gigabytes (GB) or terabytes (TB). And if you really want to impress your peers, you talk about the multipetabyte (PB) estate that you manage!

However, the *size* of a drive can also refer to the diameter of the drive's platter, which is normally referred to as the drive's *form factor*. In modern disk drives, there are two popular HDD form factors:

- 3.5-inch
- 2.5-inch

As you might expect, a drive with a 2.5-inch platter is physically smaller than a drive with a 3.5-inch platter. And as you may also expect, more often than not, a 3.5-inch drive can store more data than a 2.5-inch drive. This is simple physics; a 3.5-inch platter has more surface area on which to record data.

All 3.5-inch drives come in the same-size casing, meaning that all 3.5-inch drives can fit in the same slots in a server or disk array. The same goes for 2.5-inch drives. All 2.5-inch drives come in the same-size casing, meaning that any 2.5-inch drive can be swapped out and replaced by another. So even though technically speaking, 2.5-inch and 3.5-inch refer to the diameter of the platter, more often than not when we refer to the physical dimensions of one of these drives, we are referring to the size of the casing that it comes in. Table 2.1 shows the measurements of each of these two form factors.

TABLE 2.1 HDD form factors (approx.)

Form Factors	Height	Width	Depth
3.5-inch	26 mm (1 inch)	101 mm (4 inches)	146 mm (5.75 inches)
2.5-inch	15 mm (0.6 inch)	70 mm (2.75 inches)	100 mm (3.94 inches)

Interestingly, solid-state drive manufacturers have been quick to adopt the common 2.5-inch and 3.5-inch form factors. This can be good, because it makes it as simple as possible to use solid-state drive (SSD) instead of mechanical disk.

TIP

Not that you will ever need to know this, but, 3.5-inch and 2.5-inch drives don't actually have platters of exactly those sizes. For example, a 3.5-inch hard disk usually has a slightly larger platter (about 3.7 inches), but it is named 3.5-inch because it fits into the drive bay of an old 3.5-inch floppy drive. High-performance 10K and 15K drives often have smaller platters than their 7.2K brethren in order to keep power consumption down, as it costs more power to spin a large platter so fast. But only a geek would appreciate knowledge like that, so be careful who you tell.

Usable Capacity

While speaking about capacity, it is worth pointing out that you don't typically get what it says on the tin when you buy a disk drive. For example, when buying a 900 GB drive, you almost certainly won't get 900 GB of usable space from that drive. There are several reasons for this.

One reason is that storage capacities can be expressed as either base 10 (decimal) or base 2 (binary). As an example, 1 GB is slightly different in base 10 than in base 2:

Base 10 = 1,000,000 bytes

Base 2 = 1,048,576 bytes

NOTE

To be fair to the vendors on this topic, they use the decimal form of gigabyte and terabyte because most of their customers don't know or care about binary numbering and are familiar with the more common use of terms such as kilogram, which are also based on decimal numbering.

As you can imagine, when we're talking about a lot of gigabytes, or even terabytes, the difference between the two numbers can become significant:

Base-10 terabyte = 1,000,000,000,000 bytes

Base-2 terabyte = 1,099,511,627,776 bytes

In this TB example, the difference is just over 9 GB!

If the drive manufacturer quotes in base 2 but your OS quotes in base 10, you can see how you might feel as though you've been shortchanged.

Other factors play into this as well. For example, when formatting a disk in an OS, the OS and filesystem often consume some space as overhead. Similarly, when installing a disk in a storage array, the array will normally shave off some capacity as overhead.

There is also RAID and other things to consider. The bottom line is, don't be surprised if you get slightly less than you bargained for.

Drive Speed

Now that you know what capacity and form factor refer to, let's move on to drive speed. *Drive speed* is all about how fast the platters spin—and they spin really fast! But instead of a metric such as miles per hour, drive speed is expressed as *revolutions per minute (RPM)*. Simply put, this is how many times the platter spins every minute.

The common drive speeds are as follows:

- 5,400 RPM
- 7,200 RPM
- 10,000 RPM
- 15,000 RPM

These are usually shortened to 5.4K, 7.2K, 10K, and 15K, respectively.

NOTE

Earlier I mentioned that drive platters are rigid. This is extremely important when we look at rotational velocities. If the platter isn't stiff enough, the outer edges can experience wobble at extreme velocities such as 15,000 RPM. *Wobble* is exactly what it sounds like: movement. Any wobble or unpredictable movement will almost certainly cause a head crash and data loss on that drive.

If RPM doesn't do it for you, look at it like this: a 3.5-inch platter spinning at 15,000 RPM is moving at over 150 mph (or about 240 kilometers per hour) at the outer edges. That's what I'm talking about!

RPM converts directly into drive performance. More RPMs = more performance (and higher price, of course).

There is also a strong correlation between RPM and capacity. Usually, the higher the RPM of a drive, the lower the capacity of that drive. Conversely, the lower the RPM of a drive, the higher the capacity.

Another point worth noting is that 10K and 15K drives usually implement the SCSI command set and have a SAS or FC interface. So, faster drives implement the more performance-tuned protocol and interface. That makes sense. Also, the 5.4K and 7.2K drives usually implement the ATA command set and have a SATA interface.

If you've been reading this from the start, the following should now make perfect sense:

2.5-inch, 900 GB, 10K SAS

3.5-inch, 4 TB, 7,200 RPM SATA

TIP

Don't expect to see disks with an RPM faster than 15K. The emergence of solid-state media and the difficulty and increased costs required to develop disks with RPMs higher than 15K make it extremely unlikely that we will ever see a disk that spins faster than 15K. In fact, trends show that 10K is gaining popularity over 15K because of its lower cost and easier development for vendors.

Now that you're familiar with the anatomy and major characteristics of disk drives, let's look at how everything comes together in the performance of a drive.

Disk Drive Performance

Several factors influence disk drive performance, and the following sections cover the more important ones. We've already touched on some of these, but we will go in to more detail now.

Two major factors influence disk drive performance:

- Seek time
- Rotational latency

Both of these fall into the category of *positional latency*—basically, anything that requires the heads to move or wait for the platter to spin into position. Because these are mechanical operations, the delays that these operations introduce can be significant.

Let's take a closer look at each of them before moving on to some of the other disk drive performance considerations, such as IOPS and transfer rate.

Seek Time

Seek time is the time taken to move the read/write heads to the correct position on disk (positioning them over the correct track and sector). Faster is better, and as seek time is expressed in milliseconds (ms), the lower the number, the better.

NOTE

A *millisecond* is one-thousandth of a full second and is expressed as *ms*. In computing terms, a millisecond can seem like forever! A lot of computational operations are measured in microseconds (millionths of a second) or sometimes even nanoseconds (billionths of a second). For example, memory access on a 3 GHz processor can be 1 ns. So compared to these, a millisecond is a very long time indeed.

Seek time can therefore have a significant impact on random workloads, and excessive seeking is often referred to as *head thrashing* or sometimes just *thrashing*. And thrashing absolutely kills disk drive performance!

On very random, small-block workloads, it is not uncommon for the disk to spend over 90 percent of its time seeking. As you can imagine, this seriously slows things down! For example, a disk may achieve along the lines of 100 MBps with a nice sequential workload, but give it a nasty random workload, and you could easily be down to less than a single MBps!

TIP

More often than not, the quickest, simplest, and most reliable way to improve small-block random-read workloads is to throw solid-state media at the problem. Replace that spinning disk that detests highly random workloads with a solid-state drive that lives for that kind of work!

Some real-world average seek times taken from the spec sheets of different RPM drives are listed here:

- 15K drive: 3.4/3.9 ms
- 7.2K drive: 8.5/9.5 ms

In this list, the first number on each line is for reads, and the second number is for writes.

Real World Scenario

Disk Thrashing

A common mistake in the real world that results in head thrashing, and therefore dire performance, is misconfigured backups. Backups should always be configured so that the backup location is on separate disks than the data being backed up.

A company stored its backups on the same disks as the data that was being backed up. By doing this, they forced the disks to perform intensive read and write operations to the same disks. This involved a lot of head movement, catastrophically reducing performance. After taking a lot of time to find the source of their performance problem, they moved the backup area to separate disks. This way, one set of platters and heads was reading the source data, while another set was writing the backup, and everything worked more smoothly.

Backing up data to separate locations is also Backup 101 from a data-protection perspective. If you back up to the same location as your source data, and that shared location goes bang, you've lost your source and your backup. Not great.

Rotational Latency

Rotational latency is the other major form of positional latency. It is the time it takes for the correct sector to arrive under the R/W head after the head is positioned on the right track.

Rotational latency is directly linked to the RPM of a drive. Faster RPM drives have better rotational latency (that is, they have less rotational latency). While seek time is significant in only random workloads, rotational latency is influential in both random and sequential workloads.

Disk drive vendors typically list rotational latency as *average latency* and express it in milliseconds. Average latency is accepted in the industry as the time it takes for the platter to make a half rotation (that is, spin halfway around). Following are a couple of examples of average latency figures quoted from real-world disk drive spec sheets:

- 15K drive = 2.0 ms
- 7.2K drive = 4.16 ms

NOTE It's worth being clear up front that all performance metrics are highly dependent on variables in the test. For example, when testing the number of IOPS that a disk is capable of making, the results will depend greatly on the size of the I/O operation. Large I/O operations tend to take longer to service than smaller I/O operations.

Understanding IOPS

IOPS is an acronym for *input/output operations per second* and is pronounced *eye-ops*. IOPS is used to measure and express performance of mechanical disk drives, solid-state drives and storage arrays. As a performance metric, IOPS is most useful at measuring the performance of random workloads such as database I/O.

Unfortunately, IOPS is probably the most used and abused storage performance metric. Vendors, especially storage array vendors, are often far too keen to shout about insanely high IOPS numbers in the hope that you will be so impressed that you can't resist handing your cash over to them in exchange for their goods. So, take storage array vendor IOPS numbers with a large grain of salt! Don't worry, though. By the end of this section, you should have enough knowledge to call their bluff.

TIP

Often with IOPS numbers, or for that matter any performance numbers that vendors give you, it's less about what they are telling you and more about what they are *not* telling you. Think of it like miles per gallon for a car. MPG is one thing, but it's altogether another thing when you realize that the number was achieved in a car with a near-empty tank, no passengers, all the seats and excess weight stripped out, specialized tires that are illegal on normal roads, a massive tail wind, and driving down a steep hill with illegal fuel in the tank! You will find that, especially with storage array vendors, their numbers are often not reflective of the real world!

So what is an I/O operation? An *I/O operation* is basically a read or a write operation that a disk or disk array performs in response to a request from a host (usually a server).

NOTE

You may find that some storage people refer to an I/O operation as an *I/O* or an *IOP*. This is quite common in verbal communications, though less frequently used in writing. The most correct term is *I/O operation*, but some people are more casual in speech and often slip into using the shortened versions when referring to a single I/O operation. Obviously, if you break down an IOP to be an *input/output operation per*, it doesn't make sense. Using *an I/O* is a little better, because the operation is more clearly implied. Regardless, you will probably hear people use these terms when they mean an I/O operation.

If only it were that simple! Unfortunately, there are several types of I/O operations. Here are the most important:

- Read
- Write
- Random
- Sequential

- Cache hit
- Cache miss

To further complicate matters, the size of the I/O operation also makes a big difference.

WARNING Whenever there is complexity, such as with IOPS, vendors will try to confuse you in the hope that they can impress you. It may be a dry subject, but unless you want to get ripped off, you need to learn this!

Read and Write IOPS

Read and write IOPS should be simple to understand. You click Save on the document you are working on, and your computer saves that file to disk via one or more *write IOPS*. Later when you want to open that document to print it, your computer reads it back from disk via one or more *read IOPS*.

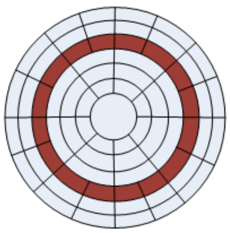
Random vs. Sequential IOPS

As you already know, mechanical disk drives thrive on sequential workloads but hunt like a lame dog when it comes random workloads. There isn't much you can do about that without getting into complicated configurations that are pointless in a world where solid state is so prevalent.

Solid-state media, on the other hand, loves random workloads and isn't at its best with sequential workloads. Let's take a quick look at a massively oversimplified, but useful, example showing why disk drives like sequential workloads so much more than random.

Figure 2.12 shows a simple overhead view of a disk drive platter. A single track (track 2, which is the third track from the outside because computers count tracks starting with 0, not 1) is highlighted with sixteen sectors (0–15).

FIGURE 2.12 Single track with 16 sectors



A sequential write to eight sectors on that track would result in sectors 0–7 being written to *in order*. This could be accomplished during a single rotation of the disk—remember that disks rotate thousands of times per minute. If that same data needed to be read back from disk, that read operation could also be satisfied by a single rotation of the disk. Not only will the disk have to rotate only once, but the read/write heads will not have to perform any seeking. So you have zero positional latency. However, there will still likely be some rotational latency while we wait for the correct sectors to arrive under the R/W heads. Nice!

Now let's assume a random workload. To keep the comparison fair, we will read or write another eight sectors' worth of data and we will stick to just track 2 as in our previous example. This time, the sectors need to be read in the following order: 5, 2, 7, 8, 4, 0, 6, 3. If the R/W head starts above sector 0, this read pattern will require five revolutions of the disk:

Rev 1 will pick up sector 5.

Rev 2 will pick up sectors 2, 7, and 8.

Rev 3 will pick up sector 4.

Rev 4 will pick up sectors 0 and 6.

Rev 5 will pick up sector 3.

This results in a lot more work and a lot more time to service the read request! And remember, that example is painfully simplistic. Things get a whole lot more complicated when we throw seeking into the mix, not to mention multiple platters!

NOTE

To be fair to disk drives and their manufacturers, they're actually a lot cleverer than in our simple example. They have a small buffer in which they maintain a queue of all I/O operations. When an I/O stream such as our random workload comes into the buffer, the I/O operations are assessed to determine whether they can be serviced in a more efficient order. In our example, the optimal read order would be 0, 2, 3, 4, 5, 6, 7, 8. While in the buffer, each I/O operation is given a tag, and each I/O operation is then serviced in tag order. Obviously, the larger the queue, the better. In our example, tagging the read commands into the optimal order would require only a single rotation.

Cache Hits and Cache Misses

Cache-hit IOPS, sometimes referred to as *cached IOPS¸* are IOPS that are satisfied from cache rather than from disk. Because cache is usually DRAM, I/O operations serviced by cache are like greased lightning compared to I/O operations that have to be serviced from disk.

When looking at IOPS numbers from large disk-array vendors, it is vital to know whether the IOPS numbers being quoted are cache hits or misses. Cache-hit IOPS will be massively higher and more impressive than cache-miss IOPS. In the real world, you will have a mixture of cache hit and cache miss, especially in read workloads, so read IOPS numbers that are purely cache hit are useless because they bear no resemblance to the real world. Statements such as *600,000 IOPS* are useless unless accompanied by more information such as *70% read, 30% write, 100% random, 4K I/O size, avoiding cache*.

Pulling the Wool over Your Eyes

In the real world, vendors are out there to impress you into buying their kits. The following are two common tricks previously employed by vendors:

Storage array vendors have been known to use well-known industry performance tools, such as Iometer, but configure them to perform ridiculously unrealistic workloads in order to make a kit look better than it is. One sleight of hand they used to play was to read or write only zeros to a system optimized to deal extremely well with predominantly zero-based workloads. The specifics are not important; the important point is that although the tool used was well-known in the industry, the workload used to measure their kit was ridiculously artificial in the extreme and bore no resemblance to any real-world workload. That latter point is extremely important!

Another old trick was to have the sample workload designed in a way that all I/O would be satisfied by cache, and the physical disk drives never used. Again, these tests came up with unbelievably high IOPS figures, but bore absolutely no resemblance to the kind of workloads the kit would be expected to service in the real world.

When reading benchmark data, always compare the size of the logical volume the test was conducted on to the amount of cache in the system being tested. If the volume isn't at least twice the size of the cache, you won't see the same performance in the real world as the benchmark suggests.

The point is, be wary of performance numbers on marketing and sales materials. And if it looks too good to be true, it probably is.

Calculating Disk Drive IOPS

A quick and easy way to estimate how many IOPS a disk drive should be able to service is to use this formula:

$$1 / (x + y) \times 1{,}000$$

where x = average seek time, and y = average rotational latency

A quick example of a drive with an average seek time of 3.5 ms and an average rotational latency of 2 ms gives us an IOPS value of 181 IOPS:

$$1 / (3.5 + 2) \times 1{,}000 = 181.81$$

This works only for spinning disks and not for solid-state media or disk arrays with large caches.

MBps

Megabytes per second (MBps) is another performance metric. This one is used to express the number of megabytes per second a disk drive or storage array can perform. If IOPS is best at measuring random workloads, MBps is most useful when measuring sequential workloads such as media streaming or large backup jobs.

TIP

MBps is not the same as Mbps. MBps is megabytes per second, whereas Mbps is megabits per second. 1 MBps is equal to 1,000 kilobytes per second.

Maximum and Sustained Transfer Rates

Maximum transfer rate is another term you'll hear in reference to disk drive performance. It is usually used to express the absolute highest possible transfer rate of a disk drive (throughput or MBps) under optimal conditions. It often refers to *burst rates* that cannot be sustained by the drive. Take these numbers with a pinch of salt. A better metric is *sustained transfer rate (STR)*.

STR is the rate at which a disk drive can read or write sequential data spread over multiple tracks. Because it includes data spread over multiple tracks, it incorporates some fairly realistic overheads, such as R/W head switching and track switching. This makes it one of the most useful metrics for measuring disk drive performance.

The following is an example of the differences between maximum transfer rate and sustained transfer rate from a real-world disk drive spec sheet for a 15K drive:

- Maximum: 600 MBps
- Sustained: 198 MBps

Disk Drive Reliability

There is no doubt that disk drives are far more reliable than they used to be. However, disk drives do fail, and when they fail, they tend to catastrophically fail. Because of this, you *absolutely must* design and build storage solutions with the ability to cope with disk drive failures.

It is also worthy of note that enterprise-class drives are more reliable than cheaper, consumer-grade drives. There are a couple of reasons:

- Enterprise-class drives are often made to a higher specification with better-quality parts. That always helps!
- Enterprise drives tend to be used in environments that are designed to improve the reliability of disk drives and other computer equipment. For example, enterprise-class disk drives are often installed in specially designed *cages* in high-end disk arrays. These cages and arrays are designed to minimize vibration, and have optimized air cooling and protected power feeds. They also reside in data centers that are temperature and humidity controlled and have extremely low levels of dust and similar problems. Compare that to the disk drive in that old PC under your desk!

Mean Time between Failure

When talking about reliability of disk drives, a common statistic is *mean time between failure*, or *MTBF* for short. It is an abomination of a statistic that can be hideously complicated and is widely misunderstood. To illustrate, a popular 2.5-inch 1.2 TB 10K SAS drive states a MTBF value of *2 million hours!* This absolutely does *not* mean that you should bank on this drive running for 228 years (2 million hours) before failing. What the statistic *might* tell you is that if you have 228 of these disk drives in your environment, you could expect to have around one failure in the first year. Aside from that, it is not a useful statistic.

Annualized Failure Rate

Another reliability specification is *annual failure rate (AFR)*. This attempts to estimate the likelihood that a disk drive will fail during a year of full use. The same 1.2 TB drive in the preceding example has an AFR of less than 1 percent.

With all of this statistical mumbo jumbo out of the way, the cold, hard reality is that disk drives fail. And when they fail, they tend to fail in style—meaning that when they die, they usually take all their data with them! So you'd better be prepared! This is when things like RAID come into play. RAID is discussed in Chapter 4, "RAID: Keeping Your Data Safe."

Solid-State Media

Let's leave all that mechanical stuff behind us and move on to the 21st century. The field of solid-state media is large, exciting, and moving at an amazing pace! New technologies and innovations are hitting the street at an incredible rate. Although solid-state media, and even flash memory, have been around for decades, uptake has been held back because of capacity, reliability, and cost issues. Fortunately for us, these are no longer issues, and solid-state storage is truly taking center stage.

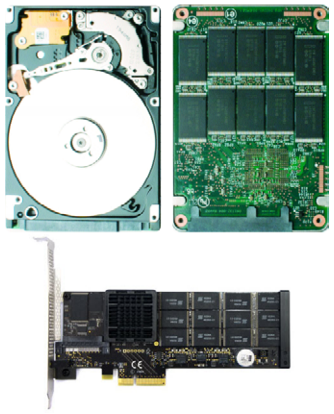
I should probably start by saying that solid-state media comes in various shapes and sizes. But these are the two that we are most interested in:

Solid-State Drive (SSD) For all intents and purposes, an SSD looks and feels like a hard disk drive (HDD). It comes in the same shape, size, and color. This means that SSDs come in the familiar 2.5- and 3.5-inch form factors and sport the same SAS, SATA, and FC interfaces and protocols. You can pretty much drop them into any system that currently supports disk drives.

PCIe Card/Solid-State Card (SSC) This is solid-state media in the form of a PCI expansion card. In this form, the solid-state device can be plugged into just about any PCIe slot in any server or storage array. These are common in servers to provide extremely high performance and low latency to server apps. They are also increasingly popular in high-end storage arrays for use as level 2 caches and extensions of array caches and tiering.

Figure 2.13 shows a mechanical drive side by side with a solid state drive, and a PCIe solid-state device beneath. The HDD and SSD are to scale, but the PCIe card is not.

FIGURE 2.13 HDD (top left), SSD (top right), and PCIe flash card (bottom)



There are also different types of solid-state media, such as flash memory, phase-change memory (PCM), memristor, ferroelectric RAM (FRAM), and plenty of others. However, this chapter focuses on flash memory because of its wide adoption in the industry. Other chapters cover other technologies when discussing the future of storage.

Unlike a mechanical disk drive, solid-state media has no mechanical parts. Like everything else in computing, solid-state devices are silicon/semiconductor based. Phew! However, this gives solid-state devices massively different characteristics and patterns of behavior than spinning disk drives. The following are some of the important high-level differences:

- Solid state is the king of random workloads, especially random reads!
- Solid-state devices give more-predictable performance because of the total lack of positional latency such as seek time.
- Solid-state media tends to require less power and cooling than mechanical disk.
- Sadly, solid-state media tends to come at a much higher $/TB capital expenditure cost than mechanical disk.

Solid-state devices also commonly have a small DRAM cache that is used as an acceleration buffer as well as for storing metadata such as the block directory and media-related statistics. They often have a small battery or capacitor to flush the contents of the cache in the event of sudden power loss.

No More RPM or Seek Time

Interestingly, solid-state media has no concept of positional or mechanical delay. There is obviously no rotational latency because there is no spinning platter, and likewise there is no seek time because there are no R/W heads. Because of this, solid-state media wipes the floor with spinning disks when it comes to random workloads. This is especially true of random reads, where solid-state performance is often closer to that of DRAM than spinning disk. This is nothing short of game changing!

Flash Memory

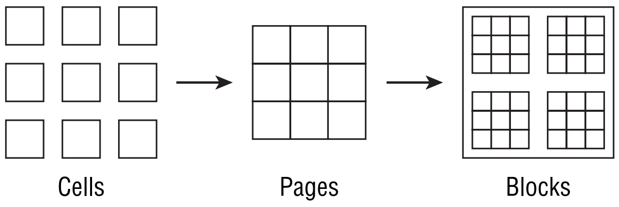
Flash memory is a form of solid-state storage. By that, I mean it is semiconductor based, has no moving parts, and provides persistent storage. The specific type of flash memory we use is called *NAND flash*, and it has a few quirks that you need to be aware of. Another form of flash memory is NOR flash, but this is not commonly used in storage and enterprise tech.

Writing to Flash Memory

Writing to NAND flash memory is particularly quirky and is something you need to make sure you understand.

Here is a quick bit of background first. Flash memory is made up of flash *cells* that are grouped into *pages*. Pages are usually 4K, 8K, or 16K in size. These pages are then grouped into *blocks* that are even bigger, often 128K, 256K, or 512K. See Figure 2.14.

FIGURE 2.14 Anatomy of flash



A brand new flash device comes with all cells set to 1. Flash cells can be programmed only from a 1 to a 0. If you want to change the value of the cell back to a 1, you need to *erase* it via a block-erase operation. Now for the small print: erasing flash memory can be done only at a block level! Yes, you read that right. Flash memory can be erased only at the *block level*. And as you may remember, blocks can be quite large. If you are new to flash, I recommend you read this paragraph again so that it sinks in.

NOTE Popular rumor has it that this block-erase procedure—in which a high voltage is applied to a whole block, and the entire contents of the block are reset to 1s—is where the term *flash* comes from. The term may also have come from older EEPROM—electronically erasable programmable read-only memory—that was erased using a UV light.

This quirky way of programming/writing to flash means a couple of important things:

- The first time you write to a flash drive, write performance will be blazingly fast. Every cell in the block is preset to 1 and can be individually programmed to 0. There's nothing more to it. It is simple and lightning quick, although still not as fast as read performance.

- If any part of a flash memory block has already been written to, *all* subsequent writes to *any* part of that block will require a much more complex and time-consuming process. This process is read/erase/program. Basically, the controller reads the current contents of the block into cache, erases the entire block, and writes the new contents to the entire block. This is obviously a much slower process than if you are writing to a clean block. Fortunately, though, this shouldn't have to occur often, because in a normal operating state a flash drive will have hidden blocks in a pre-erased condition that it will redirect writes to. Only when a flash device runs out of these pre-erased blocks will it have to revert to performing the full read/erase/program cycle. This condition is referred to as the *write cliff*.

NOTE

A *read operation* is the fastest operation that flash memory can perform. An *erase operation* is somewhere close to 10 times slower than a read. And a *program operation* is the worst of them all, being in the region of 100 times longer than a read. This is why performing the read/erase/program cycle drags the performance of flash memory down.

Why does it work like this? Economics! This process requires fewer transistors and significantly reduces manufacturing costs. And it's all about the money!

So to recap, as this may be new to you: The first time you write to a flash memory block, you can write to an individual cell or page, and this is fast. However, once you have changed a single cell in a block, any subsequent update to any cell in that block will be a lot slower because it will require a read/erase/program operation. This is a form of write amplification, which is discussed further later in the chapter.

The Anatomy of NAND Flash

Let's return to the topic of cells, pages, and blocks. This is obviously massively different from the cylinders, heads, and sectors of a spinning disk. Fortunately, flash and other solid-state controllers present the familiar LBA-style view of their contents to the wider world. This means that operating systems, drivers, and volume managers don't need to worry about these low-level differences.

You need to know about four major types of NAND flash:

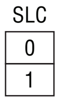
- Single-level cell (SLC)
- Multi-level cell (MLC)
- Enterprise MLC (eMLC)
- Triple-level cell (TLC)

Each has its pros and cons, as you will see.

Single-Level Cell

In the beginning was *single-level cell (SLC)*. In the SLC world, a single flash *cell* can store 1 bit of data; the electrical charge applied to the cell could be either on or off (so you can have either binary 0 or binary 1). Figure 2.15 shows an SLC flash cell.

FIGURE 2.15 SLC flash cell



Of the three flash cell options available, SLC provides the highest performance and the longest life span. However, it is also the most expensive and provides the lowest density/capacity.

NOTE Life expectancy of NAND flash is described in terms of *write endurance* or *program/erase (P/E) cycles*. SLC is usually rated at 100,000 P/E cycles per flash cell, meaning that it can be programmed (written to) and wiped 100,000 times before reliability comes into question. This is because programming and erasing the contents of a flash cell causes the physical cell to wear away.

As a result of SLC's high performance and good write endurance, it is well suited for high-end computing applications. The sad thing is, the world doesn't care too much about the high-end computing market. It isn't big enough to make a lot of money from. The world cares about consumer markets. As a result, there is less and less SLC flash out there. And as it has the highest $/GB cost, maybe it's not so bad that it is going away. Although it is true that SLC is the least power hungry of the flash cell options, this isn't a significant factor in the real world.

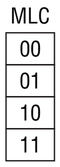
Multi-Level Cell

Multi-level cell (MLC) stores 2 bits of data per flash cell (00, 01, 10, or 11), allowing it to store twice as much data per flash cell as SLC does. Brilliant!

Well, it is brilliant from a capacity perspective at least. It is less brilliant from a performance and write endurance perspective. Transfer rates are significantly lower, and P/E cycles are commonly in the region of 10,000 rather than the 100,000 you get from SLC. However, from a random-read performance perspective, it is still way better than spinning disk! MLC has trampled all over the SLC market to the point where MLC is predominant in high-end computing and storage environments.

Interestingly, MLC is able to store 2 bits per flash cell by applying different levels of voltage to a cell. So rather than the voltage being high or low, it can be, high, medium-high, medium-low, or low. This is shown in Figure 2.16.

FIGURE 2.16 MLC flash cell



Enterprise-Grade MLC

Enterprise-grade MLC flash, known more commonly as *eMLC*, is a form of MLC flash memory with stronger error correction, and therefore lower error rates, than standard MLC flash.

Flash drives and devices based on eMLC technology also tend to be heavily overprovisioned and sport a potentially higher spec flash controller and more intelligence in the firmware that runs on the controller. As an example, a 400 GB eMLC drive might actually have 800 GB of eMLC flash but be hiding 400 GB in order to provide higher performance and better overall reliability for the drive.

On the performance front, eMLC uses the technique of overprovisioning (discussed later in the chapter) to redirect write data to pre-erased cells in the hidden area. These cells can be written to faster than cells that would have to go through the read/program/erase cycle.

On the reliability front, the eMLC drive is able to balance write workload over 800 GB of flash cells rather than just 400 GB, halving the workload on each flash cell and effectively doubling the life of the drive.

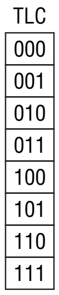
Triple-Level Cell

Triple-level cell (TLC) is 100 percent consumer driven, and can be made useful only in enterprise-class computing environments via highly sophisticated methods of smoke and mirrors. Applying complicated firmware and providing massive amounts of hidden capacity creates the illusion that the underlying flash is more reliable than it really is.

I'm sure you've guessed it: TLC stores 3 bits per cell, giving the following binary options: 000, 001, 010, 011, 100, 101, 110, 111. And as you might expect, it also provides the lowest throughput and lowest write endurance levels. Write endurance for TLC is commonly around 5,000 P/E cycles or lower.

So, TLC is high on capacity and low on performance and reliability. That is great for the consumer market, but not so great for the enterprise tech market. Figure 2.17 shows a TLC flash cell.

FIGURE 2.17 TLC flash cell



Real World Scenario

Flash Failure

Although as a professional you will need to know about reliability of media, in reality the manufacturer or vendor should be underwriting this reliability via maintenance contracts. Let's look at a couple of examples.

Storage array vendors might provide you with MLC or TLC flash devices for use in their arrays. The maintenance contracts for these arrays should ensure that any failed components are replaced by the vendor. As such you don't need to worry about reliability other than making sure that your design takes into account that devices will fail and can cope with such failures seamlessly—for example, via RAID. You shouldn't care that an SSD in one of your arrays has been replaced twice in the last year; the vendor will have replaced it for you twice at no extra cost.

In the PCIe flash card space, while failure of flash cards might well be covered under warranty or maintenance contracts, these are less likely to be RAID protected. This means that although failed cards will be replaced, your service might be down or have severely reduced performance until the failed part is replaced. In configurations like this, solid disaster recovery and High Availability (HA) clusters are important, as it may take 4–24 hours for these parts to be replaced.

Flash Overprovisioning

You might come across the terms *enterprise-grade flash drives* and *consumer-grade flash drives*. Generally speaking, the quality of the flash is the same. The major difference is usually that the enterprise stuff has a boat load of hidden capacity that the controller reserves for rainy days. This is a technique known as *overprovisioning*. A quick example is a flash drive that has 240 GB of flash capacity, but the controller advertises only 200 GB as being available. The hidden 40 GB is used by the controller for background housekeeping processes such as garbage collection and static-wear leveling. This improves performance as well as prolongs the life of the device. In these situations, the drive vendor should always list the capacity of the drive as 200 GB and not 240 GB.

Enterprise-Class Solid State

The term *enterprise-class solid state* has no standard definition. However, enterprise-class solid-state devices usually have the following over and above consumer-class devices:

- More overprovisioned capacity
- More cache
- More-sophisticated controller and firmware
- More channels
- Capacitors to allow RAM contents to be saved in the event of a sudden power loss
- More-comprehensive warranty

Garbage Collection

Garbage collection is a background process performed by the solid-state device controller. Typically, garbage collection involves cleaning up the following:

- Blocks that have been marked for deletion
- Blocks that are only sparsely filled

Blocks that have been marked for deletion can be erased in the background so that they are ready to receive writes without having to perform the dreaded read/erase/program cycle (write amplification). This obviously improves write performance.

Sparsely populated blocks can be combined into fewer blocks, releasing more blocks that can be erased and ready for fast writes. For example, three blocks that are only 30 percent occupied can be combined and reduced to a single block that will be 90 percent full, freeing up two blocks in the process.

These techniques improve performance and elongate the life of the device. However, it is a fine balance between doing too much or too little garbage collection. If you do too much, you inflict too many writes to the flash cells and unnecessarily use up precious P/E cycles for background tasks. If you do too little garbage collection, performance can be poor.

Write Amplification

Write amplification occurs when the number of writes to media is higher than the number of writes issued by the host. A quick example is when a 4K host writes to a 256K flash block that already contains data. As you learned earlier, any write to a block that already contains data requires the following procedure:

1. Read the existing contents of the block into cache.
2. Erase the contents of the block.
3. Program the new contents of the entire block.

This will obviously result in more than a 4K write to the flash media.

Write amplification slows flash performance and unnecessarily wears out the flash cells. This is to be avoided at all costs.

Channels and Parallelism

Channels are important in flash device performance. More channels = more internal throughput. If you can queue enough I/O operations to the device, then using multiple NAND chips inside the flash device concurrently via multiple channels can dramatically increase performance (parallelism). Flash drives commonly have 16- or 24-bit synchronous interfaces.

Write Endurance

Write endurance, as touched on earlier, refers to the number of program/erase (P/E) cycles a flash cell is rated at. For example, an SLC flash cell might be rated at 100,000 P/E cycles, whereas MLC might be 10,000.

The process of programming and erasing flash cells causes physical wear and tear on the cells (kind of burning or rubbing them away). Once a flash cell reaches its rated P/E cycle value, the controller marks it as unreliable and no longer trusts it.

Knowing and managing the write endurance of a solid-state device allows for the planned retirement and graceful replacement of such devices. This results in them failing catastrophically less often. For example, storage arrays with solid-state devices will monitor the P/E cycle figures of all solid-state devices and alert the vendor to proactively replace them before they wear out. EMC and HDS, and most other vendors, monitor errors on spinning disk and will replace the disks pro-actively before the disk dies. This prevents long RAID rebuilds, etc.

NOTE

Most storage arrays monitor the health of spinning disk drives and are sometimes able to predict failures and proactively replace them before they actually go pop.

Wear Leveling

Wear leveling is the process of distributing the workload across all flash cells so that some don't remain youthful and spritely, while others become old and haggard.

There are two common types of wear leveling:

- Background wear leveling
- In-flight wear leveling

In-flight wear leveling silently redirects incoming writes to unused areas of flash that have been subject to lower P/E cycles than the original target cells. The aim is to make sure that some cells aren't unduly subject to higher numbers of P/E cycles than others.

Background wear leveling, sometimes referred to as *static wear leveling*, is a controller-based housekeeping task. This process takes care of the flash cells that contain user data that doesn't change very often (these cells are usually off-limits to in-flight wear leveling). Background wear leveling is one of those tasks that has to be finely balanced so that the wear leveling itself doesn't use too many P/E cycles and shorten the life expectancy of the device.

Caching in Solid-State Devices

The storage world could not survive without caching. Even the supposedly ultra-fast solid-state devices benefit from a little DRAM cache injection (like adding nitrous oxide to your existing V8 engine).

However, aside from the obvious performance benefits cache always seems to bring, in the solid-state world, caching provides at least two important functions that help extend the life of the media:

- Write coalescing
- Write combining

Write coalescing is the process of holding writes in cache until a full page or block can be written. Assume a flash device with 128K page size. A 32K write comes in and is held in cache. Soon afterward, a 64K write comes in, and soon after that, a couple of 16K writes come in. All in all, these total 128K and can now be written to a flash block via a single program (write) operation. This is much more efficient and easier on the flash cells than if the initial 32K write was immediately committed to flash, only to then have to go through the read/erase/program cycle for each subsequent write.

Write combining happens when a particular LBA address is written to over and over again in a short period of time. If each of these writes (to the same LBA) is held in cache, only the most recent version of the data needs writing to flash. This again reduces the P/E impact on the underlying flash.

Cost of Flash and Solid State

There is no escaping it: out of the box, flash and solid-state storage comes at a relatively high $/GB cost when compared to mechanical disk. For this reason, proponents of solid-state media are keen on cost metrics that stress *price/performance*.

A popular price/performance metric is $/I/O operation. The dollar per I/O operation metric is worked out by dividing the cost of the drive by the number of IOPS it can perform. So, an SSD that does 25,000 IOPS and costs $10,000 would have a $/I/O operation rating of $0.40/I/O operation, or 40 cents per I/O operation. Compare that to a 600 GB disk drive that does 200 IOPS and costs $2,000. The disk drive will have a $/I/O operation value of $10/I/O operation. And to get 25,000 IOPS out of those 600 GB disk drives, you would need 125 of them, costing you a total of $250,000! Although this is a lot of money, 125 drives will require 125 drive bays, need power and cooling, and provide you with roughly 75 TB of capacity. This is massive overkill and very clunky if you need only 2 TB of capacity for your 10,000 IOPS! You can see why the SSD guys prefer this cost metric!

While it's not as easy to sell to your CFO, it is nonetheless a good cost metric that has value in high-performance environments.

Write Cliff

Something that you need to be aware of in the real world when looking at the performance of solid-state media is the concept of the *write cliff*, where the performance of flash memory drops significantly once each flash cell has been written to at least once.

When vendors are showing you their kit and quoting performance numbers, you need to know whether these numbers are steady-state numbers. *Steady-state numbers* (IOPS and MBps) are achieved after the device is filled up and each block has been written to at least once.

If a solid-state device is never filled to a reasonable capacity (about 70–80 percent full), you will find it hard to stress the device from a performance perspective. The device will always have enough free space to perform in-flight I/O redirection, whereby I/O is redirected to erased cells, and performance is dramatically superior than when writes have to update blocks that have not be erased.

Vendors quite often quote numbers that are not steady state and are unrealistic when compared with real-world scenarios.

Hybrid Drives

Hybrid drives are a relatively new thing. They have both a rotating platter and solid-state memory (usually flash). This is the best of both worlds, right? Well, the jury is still out, and hybrid drives are definitely not a silver-bullet solution.

Most hybrid hard drives work on a simple caching system; the most accessed sets of data get bumped up from the spinning disk area to the solid-state area of the drive, where they can be accessed quicker. That's fine for many data sets, but it has an obvious weakness when it comes to newly written data. Newly written data is typically written to spinning disk first, and then after a while of being frequently accessed, will work its way up to solid-state memory. But it usually takes time for this to happen. Some hybrid drives will try to land new data in solid state, and then tier it down if not frequently accessed. Both approaches have their pros and cons.

All in all, hybrid drives attempt to bring both capacity and speed. What goes into solid state and what goes on spinning disk is determined by the firmware on the drive. You as an admin don't have to worry about managing that process. Hybrid drives are popular in some desktop PCs and laptops but are unlikely to see high uptake in enterprise-class servers and storage arrays.

Tape

Holy cow! If spinning disk is old, magnetic tape is surely prehistoric! Yes, tape has been around for ages, and it will be around for ages to come.

Tape continues to be a popular choice for long-term storage requirements such as backup and archived data. And although it is ancient, it usually offers high capacity at a decent price point.

From a performance perspective, it's well suited for sequential access but not random access. Accessing data in random locations on tape requires the tape drive to perform multiple fast-forward and rewind operations. Not great. Chapter 11, "Backup and Recovery," discusses tape in more detail.

Further Storage Considerations

Over and above what you've already learned in this chapter, a few topics still warrant your time and attention. Let's explore these now.

Caching

The storage world loves a bit of caching, and to be honest, without caching, storage performance would be terrible. But what is caching? *Caching* is as simple as putting a small amount of memory in front of disks so that frequently accessed data can be accessed from cache memory rather than having to "go to disk." Cache memories are usually DRAM based and therefore much faster than spinning disk.

Cache memory in disk drives is so important that all enterprise-class disk drives come with a small amount of DRAM cache on board. Cache sizes of around 16 MB are fairly common on high-performance, low-capacity drives, and more in the region of 128 MB on low-performance, high-capacity drives. Although the cache is small compared to the capacity of the drive, it can be helpful with random workloads. The cache is usually protected from loss of power by a capacitor or battery.

Cache memory is also widely used in storage arrays, and it is not uncommon to see a high-end storage array sporting 1 TB of cache. While this might sound like a lot of cache, when you look closely and see that it is fronting up to 1 PB (petabyte) of disk storage, you realize that it is a relatively small amount.

Cache memory usually carries such a high $/GB cost that storage folks often use the phrase *cache is cash*, implying that cache is not cheap.

S.M.A.R.T.

Self-Monitoring, Analysis, and Reporting Technology, often referred to as SMART or S.M.A.R.T., is an industry-wide monitoring standard for hard disk drives that attempts to predict failures.

Predicting a drive failure allows you to do something about it and avoid the additional work required when a drive actually fails. If you can predict that a drive is about to fail, you can simply copy its contents to another disk known as a hot-spare. However, if the drive fails, its contents have to be rebuilt from parity. Rebuilding from parity is computationally intensive and takes a long time when compared a straight data copy.

The specifics of S.M.A.R.T., and how good it is or isn't, are not important to storage administration or design. It is one of those technologies that silently goes about its business and hopefully helps with the smooth operating of your environment.

Fragmentation

My PC is running slow. I know—it needs defragmenting. I'm sure you've either said that or heard another person say it. But will running a defrag speed up your PC?

Let's talk a little about fragmentation. Fragmentation has to be one of the most misunderstood phenomena in the storage world, probably because there is more than one type. This section covers filesystem, disk, and SSD fragmentation.

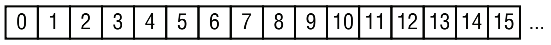
Filesystem Fragmentation

The kind of fragmentation that OS-based defrag tools, such as those found in Microsoft Windows, work to resolve is called *filesystem fragmentation*. Filesystem fragmentation occurs when files do not occupy contiguous addresses within the filesystem. That may not be a helpful description, so let's look at a quick example.

NOTE For the record, I've often heard Linux administrators say that Linux-based filesystems don't fragment data. That is not true.

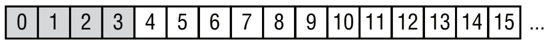
Imagine you have an empty filesystem with an address space of sixteen extents (0–15), as shown in Figure 2.18.

FIGURE 2.18 Logical filesystem address space



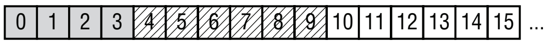
Now let's imagine you write a new file called `uber-file` that requires four extents. The filesystem will write this file out to extents 0 through 3, as highlighted by the shaded extents in Figure 2.19.

FIGURE 2.19 Partially written filesystem



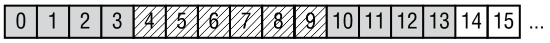
Now imagine that you write another new file, this time called `legendary-file`, which consumes six extents. The filesystem writes this file to extents 4 through 9, as shown by the diagonal lines in Figure 2.20.

FIGURE 2.20 `legendary-file` written to filesystem



Now let's assume you want to add more content to `uber-file`, doubling it in size, making it eight extents. When you save the file, another four extents will be required to save it, but there are no free extents immediately following the end of the file, and eight free contiguous extents are not available anywhere else in the filesystem. The filesystem will have to write the additional contents of `uber-file` to extents 10 through 13, as shown in Figure 2.21.

FIGURE 2.21 More content added to `uber-file`



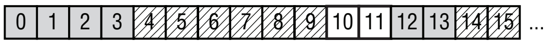
The end result is that `uber-file` is fragmented within the filesystem.

NOTE At this point, it is important to remember that the filesystem and OS have no knowledge of how the files and filesystem are laid out in the disk drive. They see only the LBA address space that the drive controller presents and have no way of ensuring that files occupy contiguous sectors and tracks. This means that any defragmentation work done by the OS or filesystem will not necessarily reorganize the on-disk layout to be more contiguous.

Defragmenting a Filesystem

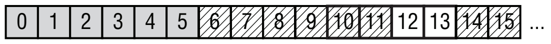
Let's stick with our sixteen-extent filesystem and pick up where we left off in Figure 2.21. To defrag this filesystem, the defrag tool will use the two free extents as a working area, or swing space, to reshuffle the extents around until files are represented in the filesystem as contiguously as possible. For example, in our simple example, you could easily shuffle extents 4 and 5 to the free extents 14 and 15. Now that 4 and 5 are free, you could move 10 and 11 to 4 and 5. After this, the filesystem would like Figure 2.22.

FIGURE 2.22 Defragmenting the filesystem



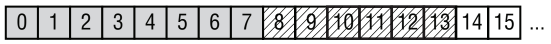
The next step would be to move extents 6 and 7 to the free extents 10 and 11. And then to move 12 and 13 to the newly freed-up extents 6 and 7. At this point, your filesystem would like Figure 2.23.

FIGURE 2.23 Continuing to defragment the filesystem



At this point, `uber-file` is no longer fragmented. You can see that moving extents 14 and 15 to 12 and 13 would lead to `legendary-file` also being unfragmented, as in Figure 2.24.

FIGURE 2.24 The defragmented filesystem



Excellent. You now have an unfragmented filesystem, and performance will no doubt be somewhat improved. But that's not the entire story! There is still disk fragmentation, which can often have a bigger impact on performance, and there is less you can do about it.

Disk Fragmentation

Disk fragmentation occurs when a file, which might be totally unfragmented in the filesystem, is fragmented on the disk. It does not occupy contiguous sectors and tracks, and as such, accessing the file incurs rotational latency and seek time that would not be necessary if the file was laid out contiguously on disk.

Disk fragmentation can naturally occur, and especially as a disk drive gets older and starts to encounter bad blocks that have to be remapped. Let's assume you have an 8K write to disk and there are sixteen free contiguous sectors (each 512 bytes). Ordinarily, an 8K write to this disk would result in the data being written to those sixteen contiguous sectors. However, if when writing the data to disk, one of these sectors is found to be bad—that is, data cannot be written to it—that sector will have to be remapped to elsewhere on disk. What you end up with is an 8K file written to fifteen contiguous sectors and a single noncontiguous sector in some other random location on disk. This means that reading or writing that file requires additional seek time and additional rotations. Shame!

And as disk drives get older, they have to remap more and more bad blocks. This is at least one of the reasons that disk drives, and computers, seem to get slower with age. And defragmenting your filesystem won't do much to alleviate this kind of fragmentation. When this happens, the best thing you can do is get a new disk drive or a new computer.

Why You Should Not Defrag an SSD

Generally speaking, you shouldn't defrag data on solid-state drives. There are a few reasons for this:

- Defragmenting a filesystem causes a lot of reading and writing. Unnecessary writing of data to solid-state media is not recommended because every write eats away at the life of your flash cells. Let's not forget that each flash cell can be programmed/erased only a finite number of times. Regularly defragging a flash device will quickly eat through that finite number of cycles and severely reduce the working life of your device. Many modern operating systems won't allow you to defrag a flash drive.
- Defragmenting solid-state drives can mess up what the controller has done. Remember that controllers on solid-state devices work in the background to optimize data layout so that future writes can avoid the effects of write amplification and other problems.
- Solid-state devices don't have moving parts, so they don't suffer from mechanical and positional latency such as seek time. A fragmented layout doesn't impact performance as badly it does for spinning disk drives.

Storage Costs

There are several things to consider when it comes to costs. Two general types of costs are involved with solid-state media and mechanical disk. You need to consider the purchase price of the storage. You also need to consider the operating costs of using the storage. To do all of that, you need to know how to measure the costs.

Acquisition Costs

The most common cost metric used when buying storage is the dollar per terabyte ($/TB) or dollar per gigabyte ($/GB). This measures only a small portion of cost—acquisition cost, and even then only against capacity with no consideration for performance. However, it is so simple and well understood that it is still the most widely used cost metric when making storage purchases. A simple example is a 100 GB SSD (or disk drive) that costs $1,000. This drive would have a $10 per gigabyte acquisition cost. If that same 100 GB drive cost $5,000, it would have a $/GB cost of $50 per gigabyte. Cost/capacity: it couldn't be simpler.

Generally speaking, solid-state media is inferior at $/TB when compared to spinning disk. As a result, the solid-state vendors are pushing another acquisition cost metric: dollars per I/O operation ($/I/O operation). This metric measures cost against performance, and in this race, the solid-state device cleans up against the spinning disk. Unfortunately for the solid-state vendors, $/I/O operation is less widely used and less well understood, partly because I/O operations are not as easy to understand and explain to a financial director (FD) as straight capacity.

Operating Costs

Solid-state media tends to cost a lot less to run/operate than mechanical disk because there are no moving parts. So there's no need for power to spin platters and move the heads, and much less heat generated as a result. In large IT environments, power, heat, and cooling can contribute significantly to data center operating expenses. This means that deploying solid-state media can reduce the overall TCO of a server, storage array, or data center.

Measuring Cost

There are few ways to measure cost when it comes to storage. Capital expenditures (cap-ex) and operating expenditures (op-ex) are the obvious ways. When saying that a higher RPM drive usually costs more than a lower RPM drive, we are referring to both cap-ex and op-ex costs. On the cap-ex front, higher RPM drives have a higher $/GB ratio, meaning that you pay more per gigabyte. On the op-ex front, making a platter spin at 10K or 15K takes a whole lot more energy than making it spin at a mere 5.4K. And the 15K drive will kick out a whole lot more heat than the 5.4K drive. Both of these can be significant in a large data center.

Over and above these factors, storage array vendors have been known to rub salt into the wounds by charging higher maintenance fees for higher-performance drives. You can end up paying through the teeth for high performance.

Summary

In this chapter we discussed in depth the anatomy and inner workings of mechanical and solid-state drives, in order to provide a solid foundation to build the rest of your storage knowledge on. We discussed the various aspects of mechanical and solid state drive technologies that impact performance, capacity, and reliability. We covered the importance of cache when it comes to performance; discussed self-monitoring analysis, reporting technologies, and fragmentation. We concluded the chapter by touching on storage costs.

Chapter Essentials

Disk Drive Performance Disk drives excel at servicing sequential workloads but struggle with heavily random workloads. Common approaches to increasing the number of IOPS a server or storage array can handle have often included adding more disk drives. More-modern approaches include adding solid-state media.

Disk Drive Reliability Although disk drives are far more reliable than they have ever been, they still fail! Make sure you design your storage environment in a way to deal with disk drive failures.

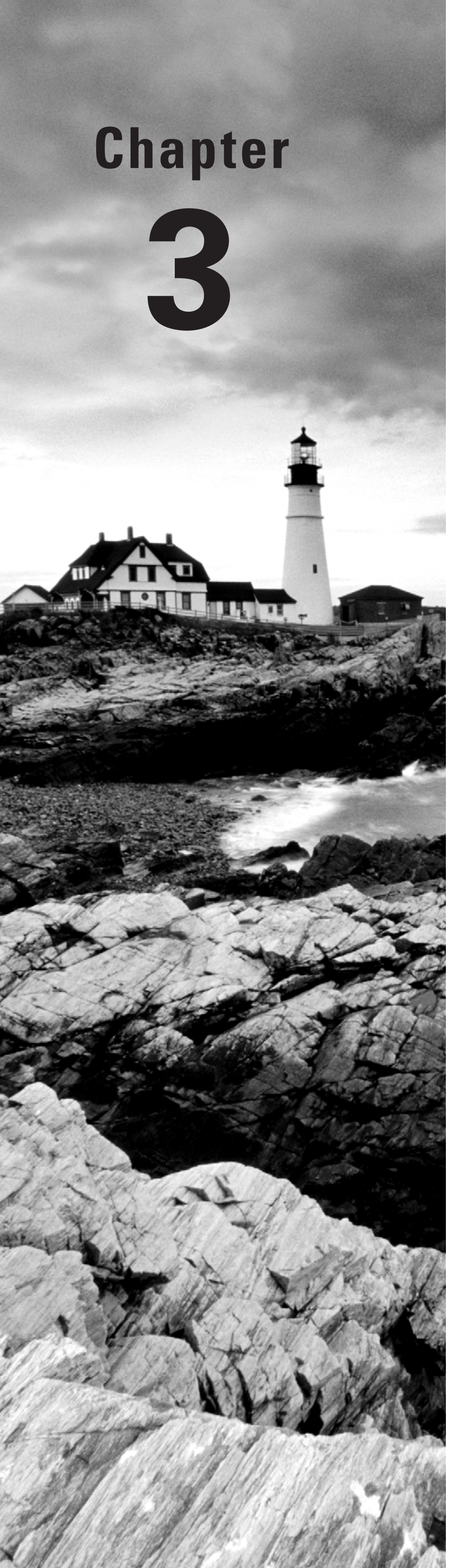
Solid-State Device Performance Solid-state devices are amazingly fast at random workloads, especially random-read workloads, where their performance is closer to that of DRAM than it is to spinning disk. However, solid-state device write performance can significantly slow down if there is not enough free space on the device for background tasks such as garbage collection to run.

Storage Performance Metrics It is vital to understand the background and detail behind any published performance metric. As with all performance metrics, they are only useful and valid to the extent that they represent real-world scenarios.

Chapter

3

Storage Arrays



TOPICS COVERED IN THIS CHAPTER:

- ✓ Enterprise-class and midrange storage arrays
- ✓ Block storage arrays and NAS storage arrays
- ✓ Flash-based storage arrays
- ✓ Storage array architectures
- ✓ Array-based replication and snapshot technologies
- ✓ Thin provisioning
- ✓ Space efficiency
- ✓ Auto-tiering
- ✓ Storage virtualization
- ✓ Host-integration technologies

This chapter builds on the previous foundational knowledge covered in Chapter 2, "Storage Devices," and shows how various drive technologies are pooled together and utilized in storage arrays. It covers the major types of storage arrays used by most businesses worldwide, including Storage Area Network (SAN) and Network Attached Storage (NAS) arrays. You will learn how they integrate with host operating systems, hypervisors, and applications, as well as about the advanced features commonly found in storage arrays. These features include technologies that play a significant role in business continuity planning, such as remote replication and local snapshots. They also include space-efficiency technologies that help improve storage utilization and bring down the cost of running a storage environment—technologies such as thin provisioning, auto-tiering, deduplication, and compression.

Storage Arrays—Setting the Scene

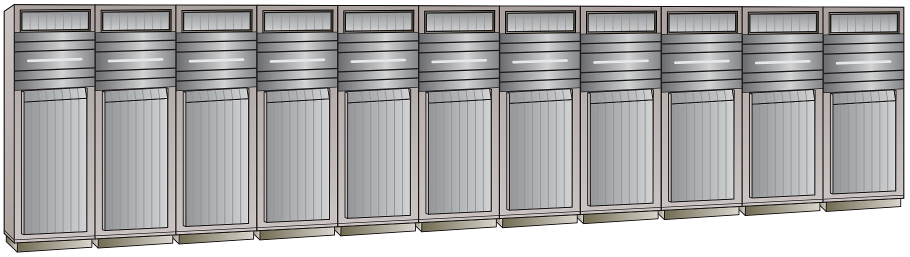
Q. Holy cow! What's that huge old piece of tin in the corner of the data center?

A. That's the storage array!

Traditionally, storage arrays have been big, honking frames of spinning disks that took up massive amounts of data-center floor space and sucked enough electricity to power a small country. The largest could be over 10 cabinets long, housing thousands of disk drives, hundreds of front-end ports, and terabytes of cache. They tended to come in specialized cabinets that were a data-center manager's worst nightmare. On top of that, they were expensive to buy, complicated to manage, and about as inflexible as granite. That was excellent if you wanted to make a lot of money as a consultant, but not so good if you wanted to run a lean and efficient IT shop that could respond quickly to business demands.

Figure 3.1 shows an 11-frame EMC VMAX array.

FIGURE 3.1 An 11-frame EMX VMAX storage array



Thankfully, this is all changing, in no small part because of the influence of solid-state storage and the cloud. Both technologies are hugely disruptive and are forcing storage vendors to up their game. Storage arrays are becoming more energy efficient, simpler to manage, more application and hypervisor aware, smaller, more standardized, and in some cases cheaper!

Here is a quick bit of terminology before you get into the details: the terms *storage array*, *storage subsystem*, *storage frame*, and *SAN array* are often used to refer to the same thing. While it might not be the most technically accurate, this book predominantly uses the term *storage array* or just *array*, as they're probably the most widely used terms in the industry.

NOTE You will sometimes hear people referring to a *storage array* as a *SAN*. This is outright wrong. A SAN is a dedicated storage *network* used for connecting storage arrays and hosts.

On to the details.

What Is a Storage Array?

A *storage array* is a computer system designed for and dedicated to providing storage to externally attached computers, usually via a storage network. This storage has traditionally been spinning disk, but we are seeing an increasing number of solid-state media in storage arrays. It is not uncommon for a large storage array to have more than a petabyte (PB) of storage.

Storage arrays *connect* to host computers over a shared network and typically provide advanced reliability and enhanced functionality. Storage arrays come in three major flavors:

- SAN
- NAS
- Unified (SAN and NAS)

SAN storage arrays, sometimes referred to as *block storage arrays*, provide connectivity via block-based protocols such as Fibre Channel (FC), Fibre Channel over Ethernet (FCoE), Internet Small Computer System Interface (iSCSI), or Serial Attached SCSI (SAS). Block storage arrays send low-level disk-drive access commands called SCSI command descriptor blocks (CDBs) such as `READ block`, `WRITE block`, and `READ CAPACITY` over the SAN.

NAS storage arrays, sometimes called *filers*, provide connectivity over file-based protocols such as Network File System (NFS) and SMB/CIFS. File-based protocols work at a higher level than low-level block commands. They manipulate files and directories with commands that do things such as create files, rename files, lock a byte range within a file, close a file, and so on.

NOTE Although the protocol is actually Server Message Block (SMB), more often than not it is referred to by its legacy name, Common Internet File System (CIFS). It is pronounced *sifs*.

Unified storage arrays, sometimes referred to as *multiprotocol arrays*, provide shared storage over both block and file protocols. The best of both worlds, right? Sometimes, and sometimes not.

The purpose of all storage arrays, SAN and NAS, is to pool together storage resources and make those resources available to hosts connected over the storage network. Over and above this, most storage arrays provide the following advanced features and functionality:

- Replication
- Snapshots
- Offloads
- High availability and resiliency
- High performance
- Space efficiency

While there are all kinds of storage arrays, it is a nonfunctional goal of every storage array to provide the ultimate environment and ecosystem for disk drives and solid-state drives to survive and thrive. These arrays are designed and finely tuned to provide an optimally cooling airflow, vibration dampening, and a clean protected power supply, as well as performing tasks such as regular scrubbing of disks and other health checking. Basically, if you were a disk drive, you would want to live in a storage array!

NOTE

SCSI Enclosure Services (SES) is one example of the kinds of services storage arrays provide for disk drives. SES is a background technology that you may never need to know about, but it provides a vital service. SES monitors power and voltages, fans and cooling, midplanes, and other environmentally related factors in your storage array. In lights-out data centers, SES can alert you to the fact that you may have a family of pigeons nesting in the warm backend of your tier 1 production storage array.

SAN Storage

As noted earlier, SAN storage arrays provide connectivity to storage resources via block-based protocols such as FC, FCoE, and iSCSI. Storage resources are presented to hosts as *SCSI logical unit numbers (LUNs)*. Think of a LUN as a raw disk device that, as far as the operating system or hypervisor is concerned, looks and behaves exactly like a locally installed disk drive—basically, a chunk of raw capacity. As such, the OS/hypervisor knows that it needs to format the LUN and write a filesystem to it.

While somewhat debatable, when compared to NAS storage, SAN storage arrays have historically been perceived as higher performance and higher cost. They provide all the snapshot and replication services that you would also get on a NAS array, but as SAN storage arrays do not own or have any knowledge of the filesystem on a volume, in some respects they are a little dumb compared to their NAS cousins.

NOTE One of the major reasons that SAN arrays have been viewed as higher performance than NAS arrays is because of the dedicated Fibre Channel network that is often required for a SAN array. Fibre Channel networks are usually dedicated to storage traffic, often employing cut-through switching techniques, and operate at high-throughput levels. Therefore, they usually operate at a significantly lower latency than sharing a 1 Gigabit or even 10 Gigabit Ethernet network, which are common when using a NAS array.

NAS Storage

NAS storage arrays work with files rather than blocks. These arrays provide connectivity via TCP/IP-based file-sharing protocols such as NFS and SMB/CIFS. They are often used to consolidate Windows and Linux file servers, where hosts mount exports and shares from the NAS in exactly the same way they would mount an NFS or CIFS share from a Linux or Windows file server. Because of this, hosts know that these exports and shares are not local volumes, meaning there is no need to write a filesystem to the mounted volume, as this is the job of the NAS array.

A Windows host wanting to map an SMB/CIFS share from a NAS array will do so in exactly the same way as it would map a share from a Windows file server by using a Universal Naming Convention (UNC) path such as `\\legendary-file-server\shares\tech`.

Because NAS protocols operate over shared Ethernet networks, they usually suffer from higher network-based latency than SAN storage and are more prone to network-related issues. Also, because NAS storage arrays work with files and directories, they have to deal with file permissions, user accounts, Active Directory, Network Information Service (NIS), file locking, and other file-related technologies. One common challenge is integrating virus checking with NAS storage. There is no doubt about it: NAS storage is an entirely different beast than SAN storage.

NAS arrays are often referred to using the NetApp term *filers*, so it would not be uncommon to hear somebody say, "We're upgrading the code on the production filers this weekend." NAS controllers are often referred to as *heads* or *NAS heads*. So a statement such as, "The vendor is on site replacing a failed head on the New York NAS" refers to replacing a failed NAS controller.

NAS arrays have historically been viewed as cheaper and lower performance than SAN arrays. This is not necessarily the case. In fact, because NAS arrays own and understand the underlying filesystem in use on an exported volume, and therefore understand files and metadata, they can often have the upper hand over SAN arrays. As with most things, you can buy cheap, low-performance NAS devices, or you can dig a little deeper and buy expensive, high-performance NAS. Of course, most vendors will tell you that their NAS devices are low cost and high performance. Just be sure you know what you are buying before you part with your company's hard-earned cash.

Unified Storage

Unified arrays provide block- and file-based storage— SAN and NAS. Different vendors implement unified arrays in different ways, but the net result is a network storage array that allows storage resources to be accessed by hosts either as block LUNs over block protocols or as network shares over file-sharing protocols. These arrays work well for customers with applications that require block-based storage, such as Microsoft Exchange Server, but may also want to consolidate file servers and use a NAS array.

Sometimes these so-called unified arrays are literally a SAN array and a NAS array, each with its own dedicated disks, bolted together with a shiny door slapped on the front to hide the hodgepodge behind. Arrays like this are affectionately referred to as *Frankenstorage*. The alternative to the Frankenstorage design is to have a single array with a common pool of drives running a single microcode program that handles both block and file.

There are arguments for and against each design approach. The Frankenstorage design tends to be more wasteful of resources and harder to manage, but it can provide better and more predictable performance, as the SAN and NAS components are actually air-gapped and running as isolated systems if you look under the hood. Generally speaking, bolting a SAN array and a NAS array together with only a common GUI is not a great design and indicates legacy technology.

Storage Array Architecture Whistle-Stop Tour

In this section, you'll examine the major components common to most storage array architectures in order to set the foundation for the rest of this chapter.

Figure 3.2 shows a high-level block diagram of a storage array—SAN or NAS—outlining the major components.

Starting from the left, we have *front-end ports*. These connect to the storage network and allow hosts to access and use exported storage resources. Front-end ports are usually FC or Ethernet (iSCSI, FCoE, NFS, SMB). Hosts that wish to use shared resources from the storage array must connect to the network via the same protocol as the storage array. So if you want to access block LUNs over Fibre Channel, you need a Fibre Channel host bus adapter (HBA) installed in your server.

If we move to the right of the front-end ports, we hit the *processors*. These are now almost always Intel CPUs. They run the array firmware and control the front-end ports and the I/O that comes in and out over them.

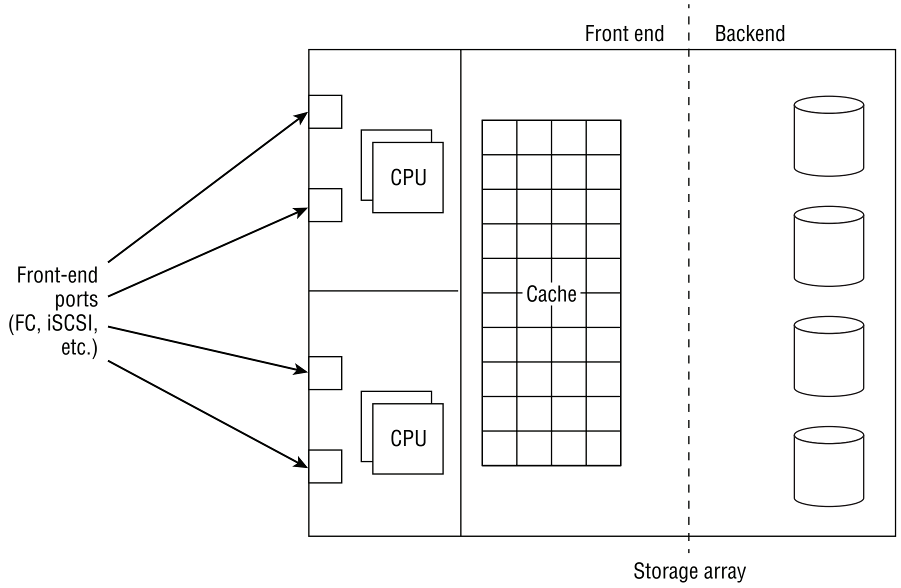
If we keep moving right, behind the processors is *cache memory*. This is used to accelerate array performance, and in mechanical disk-based arrays is absolutely critical to decent performance. No cache, no performance!

Once leaving cache, you are into the realm of the *backend*. Here there could be more CPUs, and ports that connect to the drives that compose the major part of the backend. Sometimes the same CPUs that control the front end also control the backend.

NOTE

Storage array architecture is discussed in greater detail in the "The Anatomy of a Storage Array" section later in this chapter.

FIGURE 3.2 Basic storage array block diagram



Architectural Principles

Now let's look at some of the major architectural principles common to storage arrays.

Redundancy

Redundancy in IT designs is the principle of having more than one of every component so that *when* (not *if*) you experience component failures, your systems are able to stay up and continue providing service. Redundant designs are often *N+1*, meaning that each component (N) has one spare component that can take on the load if N fails. Redundant IT design is the bedrock for *always-on enterprises* in the digital age, where even small amounts of downtime can be disastrous for businesses.

Different storage arrays are built with different levels of redundancy—and you get what you pay for. Cheap arrays will come with minimal redundant parts, whereas enterprise-class arrays will come with redundancy built in to nearly every component—power supplies, front-end ports, CPUs, internal pathways, cache modules, drive shelves, and drives. Most of these components are hot-swappable. The end goal is for the array to be able to roll with the punches and continue servicing I/O despite suffering multiple component failures.

NOTE

Hot swap refers to the ability to replace physical components within a computer system without needing to power it down. A common example is the disk drive, which should always be capable of being replaced while the system is up and servicing I/O.

Real World Scenario

An Example of Major Maintenance Performed with System Still Online

A company spent a long time troubleshooting an intermittent issue on the backend of one of their storage arrays. The issue was not major but was enough to warrant investigation. The vendor struggled to pin down the root cause of the issue and carried out a systematic plan of replacing multiple components that could be the root cause. They started by replacing the simplest components and worked their way through all possible components until only one component was left. The components they replaced included the following: disk drives, disk-drive magazines, fiber cables, backend ports, and Fibre Channel Arbitrated Loop (FC-AL) modules. All of these components were redundant and were replaced while the system was online. No connected hosts were aware of the maintenance, and no service impact was experienced. After replacing all these components, the fault persisted, and the midplane in a drive enclosure had to be swapped out. As this procedure required powering down an entire drive enclosure, removing all the drives, swapping out the midplane, and putting it all back together again, it was considered high risk and was carried out over a weekend. Although an entire drive enclosure was powered down, was removed, had components replaced, and then was powered back up, no service interruption was experienced. The system as a whole stayed up, and as far as connected systems were concerned, nothing was going on. It was all transparent to connected hosts and applications.

Dual-Controller Architectures

Dual-controller architectures are exactly what they sound like: storage arrays with two controllers.

NOTE *Controllers* are often referred to by other names such as *nodes* or *engines,* and commonly as *heads* in the NAS world.

In almost all dual-controller configurations, while both controllers are active at the same time, they are not truly active/active. Each LUN is *owned* by only one of the controllers. It is common for odd-numbered LUNs to be owned by one controller, while even-numbered LUNs are owned by the other controller. Only the controller that owns the LUN can read and write directly to it. In this sense, dual-controller arrays are active/passive on a per LUN basis—one controller owns the LUN and is therefore *active* for that LUN, whereas the other controller is *passive* and does not read and write directly to it. This is known as *asymmetric logical unit access (ALUA).*

If a host accesses a LUN via the non-owning controller, the non-owning controller will have to forward the request to the owning controller, adding latency to the process. If this situation persists, the ownership of the LUN (the controller that owns and can read and write directly to the LUN) should be changed. Special software installed on the host, referred to as multipathing software, helps manage this, and controller ownership is usually automatically changed by the system without user intervention.

Let's look at a quick example of a dual-controller system implementing ALUA. Say you have a dual-controller system with one controller called CTL0 and the other named CTL1. This system has 10 LUNs. CTL0 owns all the odd-numbered LUNs, and CTL1 owns all the even-numbered LUNs. When I say a controller *owns* a LUN, I mean it has exclusive read-and-write capability to that LUN. In this configuration, you can see that both controllers are active—hence the reason some people refer to them as active/active. However, only a single controller has read/write access to any LUN at any point in time.

Here is the small print: should one of the controllers fail in a dual-controller system, the surviving controller seizes ownership of the LUNs that were previously owned by the failed controller. Great—this might seem fine, until you realize that you have probably doubled the workload of the surviving controller. And if both controllers were already busy, the surviving controller could end up with more work than it can handle. This is never a good place to be! The moral of the story is to plan for failures and don't overload your systems.

Dual-controller systems have been around for years and are relatively simple to implement and cheap to purchase. As a result. they are popular in small- to medium-sized environments and are often referred to as *midrange arrays*.

Dual-Controller Shortcomings

The following are the major gotchas you should be aware of with dual-controller architectures:

- When one controller fails, the other controller assumes the workload of the failed controller. Obviously, this increases the workload of the surviving controller. It should be easy to see that a dual-controller array with both controllers operating at 80 percent of designed capacity will have performance issues if one controller fails.
- When a controller fails in a dual-controller system, the system has to go into what is known as *write-through mode*. In this mode, I/O has to be secured to the backend disk before an acknowledgment (ACK) is issued to the host. This is important, because issuing an ACK when the data is in cache (but not mirrored to the other controller because it is down) would result in data being lost if the surviving controller also failed. Because write-through mode does not issue an ACK when I/O hits cache, performance is severely impacted!
- Dual-controller architectures are severely limited in scalability and cannot have more than two controller nodes.

NOTE I've seen many performance issues arise in production environments that push dual-controller arrays to their limit, only to have the surviving controller unable to cope with the load when one of the controllers fails. On more than one occasion, I have seen email systems unable to send and receive mail in a timely manner when the underlying storage array is forced to run on a single controller. Even when the failed controller is replaced, the mail queues can take hours, sometimes days, to process the backlog and return to normal service. It is not good to be the email or storage admin/architect when this kind of thing happens!

Grid Storage Architectures

Grid storage architectures, sometimes referred to as *clustered* or *scale-out architectures*, are the answer to the limitations of dual-controller architectures. They consist of more than two controllers and are far more modern and better suited to today's requirements than legacy dual-controller architectures.

True Active/Active LUN Ownership

The major design difference when compared to the majority of dual-controller architectures is that all controllers in a grid-based array act as a single logical unit and therefore operate in a true active/active configuration. When saying *true active/active*, I mean that it's not limited to any of the ALUA shenanigans that we are restricted to with dual-controller architectures. In grid architectures, multiple nodes can own and write to any and all LUNs. This means that a host can issue I/O to a LUN over any of the available paths to the LUN it has, without adding the latency that occurs when a non-owning controller has to forward the request to the owning controller. In contrast, with dual-controller arrays, the host can issue I/O to a particular LUN only over paths to the owning controller without incurring additional latency.

NOTE The term *scale-out* means that you can independently add more nodes, which can contain CPUs, memory, and ports to the system, as well as more drives. This is different from scale-up systems, which can add only more drives.

Let's revisit our simple example from the "Dual-Controller Architectures" section, where we have 10 LUNs on our array, this time in a four-controller array. As this is a grid architecture, all four nodes can be active for, and write to, all 10 LUNs. This can significantly improve performance.

Seamless Controller Failures

Grid storage arrays can also deal more gracefully with controller failures. This is because they should not have to go into cache write-through mode—I say *should not*, because not all arrays are clever enough to do this. A four-node array with a single failed controller will still have three controllers up and working. There is no reason, other than poor design, that these surviving three controllers should not be able to continue providing a mirrored (protected) cache, albeit smaller in usable capacity, while the failed controller is replaced.

Here is a quick example: Assume each controller in our four-controller array has 10 GB of cache, for a system total of 40 GB of cache. For simplicity's sake, let's assume the cache is protected by mirroring, meaning half of the 40 GB is usable—20 GB. When a single controller fails, the total cache available will be reduced to 30 GB. This can still be mirror protected across the surviving three controllers, leaving ~15 GB of usable cache. The important point is that although the amount of cache is reduced, the system does not have to go into cache write-through mode. This results in a significantly reduced performance hit during node failures.

Also, a four-node array that loses a single controller node has its performance (CPUs, cache, and front-end port count) reduced by only 25 percent rather than 50 percent. Likewise, an eight-node array will have performance reduced by only 12.5 percent. Add this on top of not having to go into cache write-through mode, and you have a far more reliable and fault-tolerant system.

Scale-Out Capability

Grid arrays also scale far better than dual-controller arrays. This is because you can add CPU, cache, bandwidth, and disks. Dual-controller architectures often allow you to add only drives.

Assume that you have a controller that can deal with 10,000 Input/Output Operations Per Second (IOPS). A dual-controller array based on these controllers can handle a maximum of 20,000 IOPS—although if one of the controllers fails, this number will drop back down to 10,000. Even if you add enough disk drives behind the controllers to be able to handle 40,000 IOPS, the controllers on the front end will never be able to push the drives to their 40,000 IOPS potential. If this were a grid-based architecture, you could also add more controllers to the front end, so that the front end would be powerful enough to push the drives to their limit.

This increased scalability, performance, and resiliency comes at a cost! Grid-based arrays are usually considered enterprise class and sport an appropriately higher price tag.

Enterprise-Class Arrays

You will often hear people refer to arrays as being *enterprise class*. What does this mean? Sadly, there is no official definition, and vendors are prone to abusing the term. But typically, the term *enterprise class* suggests the following:

- Multiple controllers
- Minimal impact when controllers fail
- Online nondisruptive upgrades (NDUs)
- Scale to over 1,000 drives

- High availability
- High performance
- Scalable
- Always on
- Predictable performance
- Expensive!

The list could go on, but basically enterprise class is what you would buy every time if you could afford to!

Most storage vendors will offer both midrange and enterprise-class arrays. Generally speaking, grid-based architectures fall into the enterprise-class category, whereas dual-controller architectures are often considered midrange.

Indestructible Enterprise-Class Arrays

Enterprise-class arrays are the top dog in the storage array world, and it's not uncommon for vendors to use superlatives such as *bulletproof* or *indestructible* to describe their enterprise-class babies.

On one occasion, HP went to the extent of setting up one of their enterprise-class XP arrays in a lab and firing a bullet through it to prove that it was bulletproof. The bullet passed all the way through the array without causing an issue. Of course, the bullet took out only half of the array's controller logic, and the workload was specifically designed so as not to be affected by destroyed components. But nonetheless, it was a decent piece of marketing.

On another occasion, HP configured two of their XP enterprise-class arrays in a replicated configuration and blew up one of the arrays with explosives. The surviving array picked up the workload and continued to service the connected application.

Both are a tad extreme but good fun to watch on YouTube.

Some people argue that for an array to be truly enterprise class, it must support mainframe attachment. True, all mainframe-supporting arrays are enterprise class, but not all enterprise-class arrays support mainframe!

Midrange Arrays

Whereas enterprise arrays have all the bells and whistles and are aimed at customers with deep pockets, *midrange arrays* are targeted more toward the cost-conscious customer. Consequently, midrange arrays are more commonly associated with dual-controller architectures.

While midrange arrays are by no means bad, they definitely lag behind enterprise arrays in the following areas:

- Performance
- Scalability
- Availability

However, they tend to come in at significantly lower costs (for both capital expenditures and operating expenditures).

Referring to an array as midrange indicates that it is not quite low-end. Although these arrays might not be in the same league as enterprise arrays, they often come with good performance, availability, and a decent feature-set for a more palatable price.

All-Flash Arrays

As far as storage and IT administrators are concerned, *all-flash storage arrays* operate and behave much the same as traditional storage arrays, only faster! Or at least that is the aim. By saying that they behave the same, I mean that they have front-end ports, usually some DRAM cache, internal buses, backend drives, and the like. They take flash drives, pool them together, carve them into volumes, and protect them via RAID or some other similar protection scheme. Many even offer snapshot and replication services, thin provisioning, deduplication, and compression. All-flash arrays can be dual-controller, single-controller, or scale-out (and no doubt anything else that appears in the future).

However, there are subtle and significant differences under the hood that you need to be aware of. Let's take a look.

Solid-state technology is shaking up the storage industry and changing most of the rules! This means that you cannot simply take a legacy architecture that has been designed and fine-tuned over the years to play well with spinning disk, and lash it full of solid-state technology. Actually, you can do this, and some vendors have, but you will absolutely not be getting the most from your array or the solid-state technology you put in it.

Solid-state storage changes most of the rules because it behaves so differently than spinning disk. This means that all the fine-tuning that has gone into storage array design over the last 20+ years—massive front-end caches, optimized data placement, prefetching algorithms, backend layout, backend path performance, and so on—can all be thrown out the window.

Cache in All-Flash Arrays

The main purpose of cache is to hide the mechanical latencies (slowness) of spinning disk. Well, flash doesn't suffer from any of these maladies, making that large, expensive cache less useful. All-flash arrays will probably still have a DRAM cache, because as fast as flash is, DRAM is still faster. However, the amount of DRAM cache in all-flash arrays will be smaller and predominantly used for caching metadata.

Caching algorithms also need to understand when they are talking to a flash backend. There is no longer a need to perform large prefetches and read-aheads. Reading from flash is super fast anyway, and there is no waiting around for heads to move or the platter to spin into location.

Flash Enables Deduplication

Seriously. I know it might seem ridiculous that an all-flash array would be well suited to a technology that has historically been viewed as impeding performance: primary storage. But remember, flash is changing the rules.

First up, the cost per terabyte of all-flash arrays demands data-reduction technologies such as deduplication and compression in order to achieve a competitive $/TB. If implemented properly, on top of a modern architecture—one designed to work with solid-state storage—inline deduplication of primary storage could have a zero performance impact. It can work like this: Modern CPUs come with offload functions that can be used to perform extremely low-overhead lightweight hashes against incoming data. If a hash suggests that the data may have been seen before and therefore is a candidate for deduplication, a bit-for-bit comparison can be performed against the actual data. And here is the magic: bit-for-bit comparisons are lightning fast on flash media, as they are essentially read operations. On spinning disk, they were slow! Implementing inline deduplication on spinning-disk architectures required strong hashes in order to reduce the need for bit-for-bit comparisons. These strong hashes are expensive in terms of CPU cycles, so they imposed an overhead on spinning-disk architectures. This is no problem with flash-based arrays!

Also, with a moderate cache in front of the flash storage, these hashes can be performed asynchronously—after the ACK has been issued to the host.

Deduplication also leads to noncontiguous data layout on the backend. This can have a major performance impact on spinning-disk architectures, as it leads to a lot of seeking and rotational delay. Neither of these are factors for flash-based architectures.

That said, deduplication still may not be appropriate for all-flash arrays aimed at the ultra-high-performance tier 0 market. But for anything else (including tier 1), offering deduplication should almost be mandatory!

NOTE

Throughout the chapter, I use the term *tier* and the concept of tiering to refer to many different things. For example, an important line-of-business application within an organization may be referred to as a *tier 1* application, while I also refer to high-end storage arrays as *tier 1* storage arrays. I also use the term to refer to drives. Tier 1 drives are the fastest, highest-performing drives, whereas tier 3 drives are the slowest, lowest-performing drives. One final point on tiering: the industry has taken to using the term *tier 0* to refer to ultra-high-performance drives and storage arrays. I may occasionally use this term.

Flash for Virtual Server Environments

Hypervisor technologies tend to produce what is known in the industry as the *I/O blender effect*. This is the unwanted side effect of running tens of virtual machines on a single physical server. Having that many workloads running on a single physical server results in all I/O coming out of that server being highly randomized—kind of like throwing it all in a food mixer and blending it up! And we know that spinning disk doesn't like random I/O. Solid-state storage, on the other hand, practically lives for it, making all-flash arrays ideal for highly virtualized workloads.

Tier 0 or Tier 1 Performance

There is a definite split in the goal of certain all-flash arrays. There are those gunning to open up a new market, the so called tier 0 market, where uber-high performance and dollar per IOP ($/IOP) is the name of the game. These arrays are the dragsters of the storage world, boasting IOPS in the millions. On the other hand are the all-flash arrays that are aiming at the current tier 1 storage market. These arrays give higher and more-predictable performance than existing tier 1 spinning-disk arrays, but they do not give performance in the range of the I/O dragsters of the tier 0 market. Instead they focus more on features, functionality, and implementing space-saving technologies such as deduplication in order to come in at a competitive dollar per gigabyte ($/GB) cost.

Storage Array Pros

Storage arrays allow you to pool storage resources, thereby making more-efficient use of both capacity and performance. From a performance perspective, by parallelizing the use of resources, you can squeeze more performance out of your kit. This lets you make better use of capacity and performance.

Capacity

From a capacity perspective, storage arrays help prevent the buildup of *stranded capacity*, also known as *capacity islands*. Let's look at a quick example: 10 servers each with 1 TB of local storage. Five of the servers have used over 900 GB and need more storage added, whereas the other five servers have used only 200 GB. Because this storage is locally attached, there is no way to allow the five servers that are running out of capacity to use spare capacity from the five servers that have used only 200 GB each. This leads to stranded capacity.

Now let's assume that 10 TB of capacity was pooled on a storage array with all 10 servers attached. The entire capacity available can be dynamically shared among the 10 attached servers. There is no stranded capacity.

Performance

The performance benefits of storage arrays are similar to the capacity benefits. Sticking with our 10-server example, let's assume each server has two disks, each capable of 100 IOPS. Now assume that some servers need more than the 200 IOPS that the locally installed disks can provide, whereas other servers are barely using their available IOPS. With locally attached storage, the IOPS of each disk drive is available to only the server that the drive is installed in. By pooling all the drives together in a storage array capable of using them all in parallel, all of the IOPS are potentially available to any or all of the servers.

Management

Storage arrays make capacity and performance management simpler. Sticking with our 10-server example, if all the storage was locally attached to the 10 servers, there would be 10 management points. But if you are using a storage array, the storage for these servers can be managed from a single point. This can be significant in large environments with hundreds, or even thousands of servers that would otherwise need their storage individually managed.

Advanced Functionality

Storage arrays tend to offer advanced features including replication, snapshots, thin provisioning, deduplication, compression, high availability, and OS/hypervisor offloads. Of course, the cloud is now threatening the supremacy of the storage array, forcing the storage array vendors to up their game in many respects.

Increased Reliability

As long as a storage array has at least two controllers, such as in a dual-controller system, it can ride out the failure of any single component, including an entire controller, without losing access to data. Sure, performance might be impacted until the failed component is replaced; however, you will not lose your data. This cannot be said about direct-attached storage (DAS) approaches, where storage is installed directly in a server. In the DAS approach, the failure of a server motherboard or internal storage controller may cause the system to lose data.

Storage Array Cons

Although storage arrays have a lot of positives, there are some drawbacks. In this section, you'll take a look at some of them.

Latency

Latency is one area where storage arrays don't set the world alight. Even with the fastest, lowest-latency storage network, there will always be more latency reading and writing to a storage array than reading and writing to a local disk over an internal PCIe bus. Therefore, for niche use cases, where ultra-low latency is required, you may be better served with a local storage option. And if you need high random IOPS on top of low latency, then locally attached, PCIe-based flash might be the best option. However, the vast majority of application storage requirements can be met by a storage array. Only ultra-low latency requirements require locally attached storage.

Lock-in

Storage arrays can lock you in if you're not careful. You invest in a significant piece of infrastructure with a large up-front capital expenditure investment and ongoing maintenance. And within four to five years, you need to replace it as part of the technology-refresh cycle. Migrating services to a new storage array as part of a tech refresh can often be a mini project in and of itself. While technologies are emerging to make migrations easier, this can still be a major pain. Make sure you consider technology-refresh requirements when you purchase storage. Although five years might seem a long way off, when you come to refresh your kit, you will kick yourself if you have to expend a lot of effort to migrate to a new storage platform.

Cost

There is no escaping it: storage arrays tend to be expensive. But hey, you get what you pay for, right? It's up to you to drive a hard bargain with your storage vendor. If they don't offer you a good-enough price, it's a free market out there with plenty of other vendors who will be more than happy to sell you their kits. Don't make the mistake of simply trusting a salesperson and not doing your research.

The Anatomy of a Storage Array

Let's get under the hood a little now. Figure 3.3 shows the major components of a storage array (the same as previously seen in Figure 3.2).

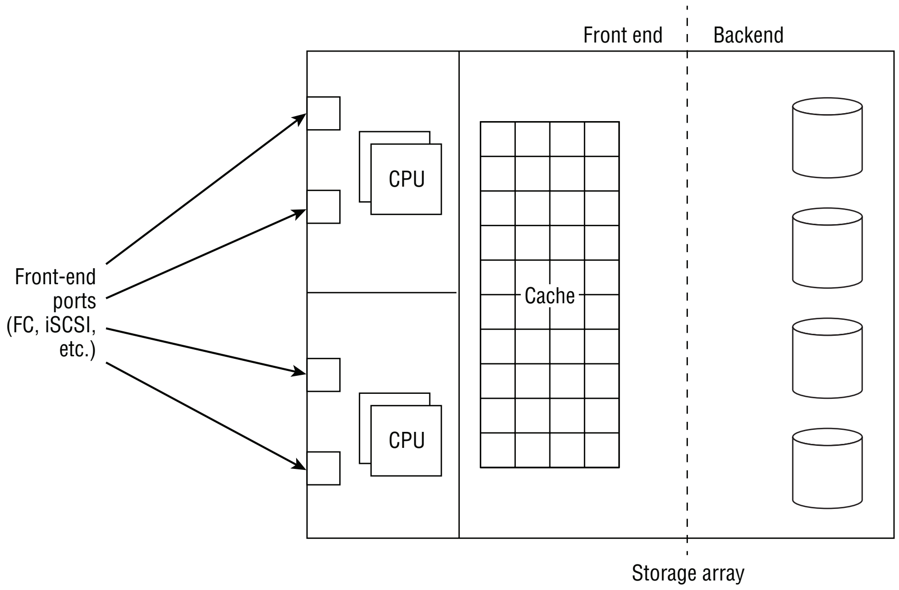
We'll start at the front end and work our way through the array, all the way to the backend (left to right on Figure 3.3).

Front End

The *front end* of a storage array is where the array interfaces with the storage network and hosts—the gateway in and out of the array, if you will. It is also where most of the magic happens.

If you know your SCSI, the front-end ports on a SAN storage array act as SCSI targets for host I/O, whereas the HBA in the host acts as the SCSI initiator. If your array is a NAS array, the front-end ports are network endpoints with IP addresses and Domain Name System (DNS) hostnames.

FIGURE 3.3 Major components of a storage array



Ports and Connectivity

Servers communicate with a storage array via ports on the front, which are often referred to as *front-end ports*. The number and type of front-end ports depends on the type and size of the storage array. Large enterprise arrays can have hundreds of front-end ports, and depending on the type of storage array, these ports come in the following flavors: FC, SAS, and Ethernet (FCoE and iSCSI protocols).

Most high-end storage arrays have hot-swappable front-end ports, meaning that when a port fails, it can be replaced with the array still online and servicing I/O.

Hosts can connect to storage arrays in different ways, such as running a cable directly from the server to the storage array in a *direct attached approach* or connecting via a network. It is also possible to have multiple connections, or *paths,* between a host and a storage array, all of which need managing. Let's look more closely at each of these concepts.

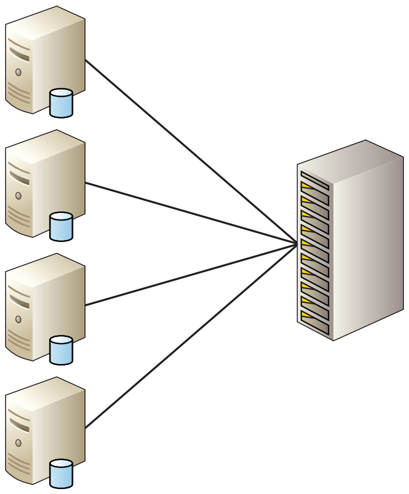
Direct Attached

Hosts can connect directly to the front-end ports without an intervening SAN switch in a connectivity mode known as *direct attached*. Direct attached has its place in small deployments but doesn't scale well. In a direct attached configuration, there can be only a one-to-one mapping of hosts to storage ports. If your storage array has eight front-end ports, you can have a maximum of eight servers attached. In fact, as most people follow the industry-wide best practice of having multiple paths to storage (each host is connected to at least two storage array ports), this would reduce your ratio of hosts to storage ports from one-to-one to one-to-two, meaning that an eight-port storage array could cater to only four hosts—each with two paths to the storage array. That is not scalable by any stretch of the imagination.

SAN Attached

Far more popular and scalable than direct attached is SAN attached. *SAN attached* places a switch between the server and the storage array and allows multiple servers to share the same storage port. This method of multiple servers sharing a single front-end port is often referred to as *fan-in* because it resembles a handheld fan when drawn in a diagram. Fan-in is shown in Figure 3.4.

FIGURE 3.4 Fan-in

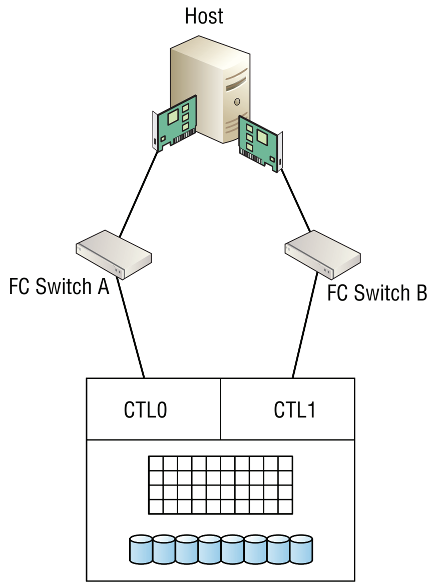


Multipath I/O

A mainstay of all good storage designs is redundancy. You want redundancy at every level, including paths across the network from the host to the storage array. It is essential that each server connecting to storage has at least two ports to connect to the storage network so that if one fails or is otherwise disconnected, the other can be used to access storage. Ideally, these ports will be on separate PCIe cards.

However, in micro server and blade server environments, a single HBA with two ports is deployed. Having a single PCIe HBA with two ports is not as redundant a configuration as having two separate PCIe HBA cards, because the dual-ported PCIe HBA is a single point of failure. Each of these ports should be to discrete switches, and each switch should be cabled to different ports on different controllers in the storage array. This design is shown in Figure 3.5.

FIGURE 3.5 Mutipath I/O



Host-based software, referred to as *multipath I/O (MPIO) software*, controls how data is routed or load balanced across these multiple links, as well as taking care of seamlessly dealing with failed or flapping links.

Common MPIO load-balancing algorithms include the following:

Failover Only Where one path to a LUN is active and the other passive, no load balancing is performed across the multiple paths.

Round Robin I/O is alternated over all paths.

Least Queue Depth The path with the least number of outstanding I/Os will be used for the next I/O.

All modern operating systems come with native MPIO functionality that provides good out-of-the-box path-failure management and load balancing. OS and hypervisor MPIO architectures tend to be frameworks that array vendors can write device-specific modules for. These device-specific modules add functionality to the host's MPIO framework—including additional load-balancing algorithms that are tuned specifically for the vendor's array.

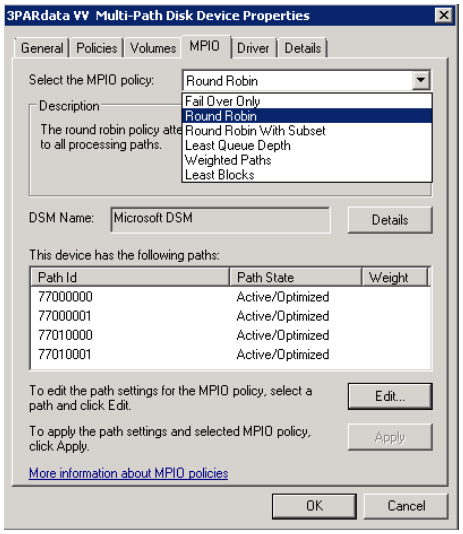
Exercise 3.1 walks you through configuring MPIO load balancing.

EXERCISE 3.1

Configuring an MPIO Load-Balancing Policy in Microsoft Windows

The following procedure outlines how to configure the load-balancing policy on a Microsoft Windows server using the Windows MPIO GUI:

1. Open the Microsoft Disk Management snap-in by typing **diskmgmt.msc** at the command line or at the Run prompt.
2. In the Disk Management UI, select the disk you wish to set the load-balancing policy for, right-click it, and choose Properties.
3. Click the MPIO tab.
4. From the Select MPIO Policy drop-down list, choose the load-balancing policy that you wish to use.


3PARdata VV Multi-Path Disk Device Properties

General | Policies | Volumes | MPIO | Driver | Details

Select the MPIO policy: Round Robin

Fail Over Only
Round Robin
Round Robin With Subset
Least Queue Depth
Weighted Paths
Least Blocks

Description

The round robin policy atte
to all processing paths.

DSM Name: Microsoft DSM

This device has the following paths:

Path Id	Path State	Weight
77000000	Active/Optimized	
77000001	Active/Optimized	
77010000	Active/Optimized	
77010001	Active/Optimized	

To edit the path settings for the MPIO policy, select a path and click Edit.

To apply the path settings and selected MPIO policy, click Apply.

More information about MPIO policies

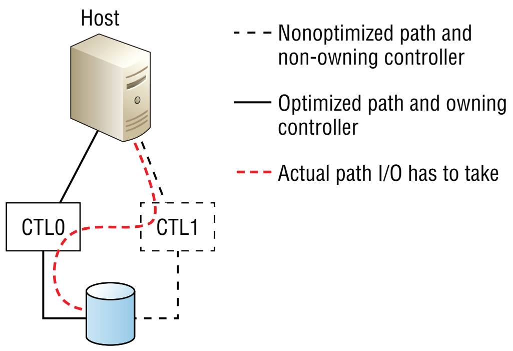
This graphic shows all four paths to this LUN as Active/Optimized, which tells us that this LUN exists on a grid architecture–based array. If it was a LUN on a dual-controller array that supported only ALUA, only one path would be Active/Optimized, and all other paths would be failover only.

The Microsoft Device Specific Module (DSM) provides some basic load-balancing policies, but a vendor-specific DSM may offer additional ones.

Some vendors also offer their own multipathing software. The best example of this is EMC PowerPath. EMC PowerPath is a licensed piece of host-based software that costs money. However, it is specifically written to provide optimized multipathing for EMC arrays. It also provides the foundation for more-advanced technologies than merely MPIO.

Using MPIO for dual-controller arrays that support only ALUA requires that multiple paths to LUNs be configured as failover only, ensuring that only the path to the active controller (the controller that owns the LUN) is used. Accessing a LUN over a nonoptimized path (a path via the controller that does not own the LUN) can result in poor performance over that path. This is because accessing a LUN over a nonoptimized path results in the I/O having to be transferred from the controller that does not own the LUN, across the controller interconnects, to the controller that does own it. This incurs additional latency. This is depicted in Figure 3.6.

FIGURE 3.6 Accessing a LUN over a nonoptimized path


Host
Nonoptimized path and non-owning controller
Optimized path and owning controller
Actual path I/O has to take
CTL0
CTL1

Front-End Port Speed

Front-end ports can usually be configured to operate at a few different speeds, depending on the type of front-end port they are. For example, a 16 Gbps FC front-end port can usually be configured to operate at 16 Gbps, 8 Gbps, or 4 Gbps. In a Fibre Channel SAN, it is always a good idea to hard-code the port speed and not rely on the autonegotiate protocol. When doing this, make sure you hard-set the ports at either end of the cable to the same speed. This will ensure the most reliable and stable configuration. However, Ethernet-based storage networks should use the autonegotiate (AN) setting and should not hard-code port speeds.

CPUs

Also key to the front end are the CPUs. CPUs, typically Intel, are used to power the front end. They execute the *firmware* (often called *microcode* by storage people) that is the brains of the storage array. Front-end CPUs, and the microcode they run, are usually responsible for all of the following:

- Processing I/O
- Caching
- Data integrity

- Replication services
- Local copy services
- Thin provisioning
- Compression
- Deduplication
- Hypervisor and OS offloads

It is not uncommon for large, enterprise-class storage arrays to have about 100 front-end CPUs in order to both increase the front-end processing power and provide redundancy. It is also common for smaller, low-end storage arrays to have very few CPUs and use the same CPUs to control the front end and the backend of an array.

Vendors are keen to refer to their storage arrays as *intelligent*. While that is a bit of a stretch and a bit of an insult to anything truly intelligent, if there is any intelligence in a storage array, that intelligence resides in the firmware that runs the array.

NOTE

As mentioned, the term *microcode* is frequently used in the storage industry to refer to the software/firmware that is the brains of a storage array. You may also see the term shortened it to *ucode.* Technically speaking, this should really be written as *μcode* (the symbol *μ* is the SI symbol for *micro*) but as the μ symbol doesn't appear on any standard QWERTY keyboard, it is easier to substitute the letter *u* for it. Long story short, the term *ucode* refers to the microcode that runs the array. It is pronounced *you-code*.

Because the firmware is the brains of the array, you'd be right in thinking that firmware management is crucial to the stability and performance of any storage array. If you don't bother keeping it up-to-date, you will fall out of vendor support and put yourself at risk when issues arise. You really can't afford to put your storage in jeopardy by not keeping your firmware up-to-date.

On the other hand, you probably don't want to be on the hemorrhaging edge of firmware technology. Running the very latest version of firmware the day it is released (or heaven forbid, a prerelease version) in your production environment massively increases your risk of encountering bugs, which at best wastes time and money, and at worst can temporarily cripple your organization. A good rule of thumb is to wait a minimum of three months after general availability (GA) before dipping your toe into the latest and greatest firmware revision from a vendor.

It is also a good idea to methodically deploy new firmware to your estate. A common approach is as follows:

1. LAB. Deploy in the lab first and run for at least a month, testing all of your configuration, including testing failure scenarios and rebooting attached servers.
2. DEV. Deploy to your development environment and run for at least a month.
3. DR. Deploy to your disaster recovery (DR) arrays (if you operate a live/DR environment) and run for one week.
4. PROD. Deploy to your live production arrays.

Obviously, this list will have to be tweaked according to your environment, as not everyone has the luxury of a fully functional lab, development environment, and so on. However, here are a few points worth noting:

- Let somebody else find the bugs. This is the main reason for allowing the code to mature in the wild for three months before you take it. Usually within three months of deployment, people have uncovered the worst of the bugs. It also allows the vendor time to patch anything major.
- Do not deploy directly to your live production environment. You want to give yourself a fighting chance of weeding out any potential issues in your less-mission-critical environments.
- If you replicate between storage arrays, it is usually not a great idea to have them running different versions of firmware for very long. It is usually a good idea to upgrade the DR array one weekend and then upgrade the live production array the following weekend. However, if you do not have a lab or development environment to soak test in first, it may be a better idea to run in DR for two to four weeks before upgrading live production arrays.

WARNING

Always check with your vendor that you are upgrading via a supported route. Some vendors will not allow you to run different versions of code on arrays that replicate to each other. The preceding rules are guidelines only, and you need to engage with your vendor or channel partner to ensure you are doing things by the book.

An obvious reason to circumvent the preceding rules is if the new version of code includes a major bug fix that you are waiting for.

Another thing to be aware of is that many organizations standardize on an annual firmware upgrade plan. Upgrading annually allows you to keep reasonably up-to-date and ensures you don't fall too far behind.

One final point worth noting on the topic of firmware management is to ensure that you understand the behavior of your array during a firmware upgrade. While most firmware upgrades these days tend to be NDUs, you need to check. The last thing you want to do is upgrade the firmware only to find out halfway through that all front-end ports will go offline part way through the upgrade, or an entire controller will go offline for a while, lowering the overall performance capability of the array during the upgrade!

LUNs, Volumes, and Shares

Disks are installed on the backend, and their capacity is carved into volumes by the array. If your storage array is a SAN array, these volumes are presented to hosts via front-end ports as LUNs. Rightly or wrongly, people use the terms *volume* and *LUN* interchangeably. To the host that sees the LUN, it looks and behaves exactly like a locally installed disk drive. If your array is a NAS array, these volumes are presented to hosts as network shares, usually NFS or SMB/CIFS.

LUNs on a storage array are usually expandable, meaning you can grow them in size relatively easily. There are sometimes a few complications such as replicated LUNs, but this shouldn't be an issue on any decent storage array. If it is, you are buying something ancient.

Reducing the size of a LUN, on the other hand, is far more dangerous. For this reason, it is almost never performed in block-storage environments where the storage array has no knowledge of the filesystem on the LUN. However, NAS arrays have the upper hand here, as they own both the filesystem and the volume. Knowledge truly is power, and it is common to see good NAS arrays offer the ability to shrink volumes.

LUN Masking

As a security precaution, all LUNs presented out of a SAN storage array should be masked on the array. *LUN masking* is the process of controlling which servers can see which LUNs. Actually, it is the process of controlling which HBAs can see which LUNs. It is basically access control. Without LUN masking, all LUNs presented out of the front of a storage array would be visible to all connected servers. As you can imagine, that would be a security and data-corruption nightmare.

These days, LUN masking is almost always performed on the storage array—using the World Wide Port Name (WWPN) of a host's HBA in FC environments, and using the IP address or iSCSI Qualified Name (IQN) in iSCSI environments. For example, on a storage array, you present LUNs on a front-end port. That port has an access control list on it that determines which host HBA WWPNs are allowed to access which LUNs. With LUN masking enabled, if your host's HBA WWPN is not on the front-end port's access control list, you will not be able to see any LUNs on that port. It is simple to implement and is standard on all storage arrays. Implement it!

Real World Scenario

The Importance of Decommissioning Servers Properly

A common oversight in storage environments is not cleaning up old configurations when decommissioning servers. Most companies will at some point have decommissioned a server and then rebuilt it for another purpose, but will have not involved the storage team in the decom process. When the server is rebuilt and powered up, it can still see all of its old storage from its previous life. This is because the server still has the same HBA cards with the same WWPNs, and these WWPNs were never cleared up from the SAN zoning and array-based LUN masking. In most cases, this is a minor annoyance, but it can be a big problem if the server used to be in a cluster that had access to shared storage!

Back in the day, people used to apply masking rules on the host HBA as well. In modern storage environments, this is rarely if ever done anymore. One of the major reasons is that it was never a scalable solution. In addition, this provided little benefit beyond performing LUN masking on the storage array and SAN zoning in the fabric. Exercise 3.2 discusses how to configure LUN masking.

EXERCISE 3.2

Configuring LUN Masking

The following step-by-step example explains a common way to configure LUN masking on a storage array. The specifics might be different depending on the array you are using. In this example, you will configure a new ESX host definition for an ESX host named `legendary-host.nigelpoulton.com`.

1. Create a new host definition on the array.
2. Give the host definition a name of **legendary-host**. This name can be arbitrary but will usually match the hostname of the host it represents.
3. Set a *host mode* or *host persona* of **Persona 11 - VMware**. This is the persona type for an ESX host on an HP 3PAR array and should be used for all ESX host definitions. The value will be different on different array technologies.
4. Add the WWPNs of the HBAs in `legendary-host` to the host definition. `legendary-host` is configured on your array, and you can now map volumes to it.

   The next steps will walk you through the process of mapping a single 2 TB volume to `legendary-host`:
5. Select an existing 2 TB volume and choose to export this volume to a host.
6. Select `legendary-host` as the host that you wish to export the volume to.

Once you have completed these procedures, a 2 TB volume will be exported to `legendary-host` (actually the HBA WWPNs configured as part of the `legendary-host` host definition in step 4). No other hosts will be able to access this volume, as the ACL on the array will allow only initiators with the WWPNs defined in step 4 to access the LUN.

The equivalent of LUN masking in the NAS world is restricting the IP address or hostname to which a volume/share is exported. Over and above that, you can also implement file and folder permissions.

LUN Sharing

While it is possible for multiple hosts to access and share the same block LUN, this should be done with extreme care! Generally speaking, the only time multiple servers should be allowed to access and write to the same shared LUN is in cluster configurations in which the cluster is running proper clustering software that ensures the integrity of the data on the LUN—usually by ensuring that only a single server can write to the LUN at any one time.

Certain backup software and designs may require the backup media server to mount a shared LUN as read-only so that it can perform a backup of the data on a LUN. But again, the backup software will be designed to do this.

If two servers write to the same LUN without proper clustering software, the data on the LUN will almost certainly become corrupt.

Exercise 3.3 examines how to provision a LUN to a cluster.

EXERCISE 3.3

LUN Presentation in a Cluster Configuration

This exercise walks you through provisioning a single 2 TB LUN to an ESX cluster that contains two ESX hosts named ESX-1 and ESX-2. These two hosts are already defined on the array.

1. Create a new host set definition on the array. A *host set* is a host group that can contain multiple host definitions.
2. Give the host set a name of **legendary-ESX-cluster**.
3. Add the ESX-1 and ESX-2 hosts to the host set. The host set `legendary-ESX-cluster` is now configured on your array and contains two ESX hosts and their respective WWPNs.

The next step is to map your 2 TB volume to the host set.

4. Select an existing 2 TB volume and choose to export this volume to a host set.
5. Select `legendary-ESX-cluster` as the host set that you wish to export the volume to.

The 2 TB volume is now presented to ESX hosts ESX-1 and ESX-2. Both ESX hosts can now access this volume.

Thick and Thin LUNs

LUNs on a storage array can be thick or thin. Traditionally, they were always thick, but these days, more often than not they are thin. We will discuss thick and thin LUNs later in the chapter, in the "Thin Provisioning" section.

Cache

It could be said that the *cache* is the heart of a storage array. It is certainly the Grand Central Station of an array. On most storage arrays, everything has to pass through cache. All writes to the array go to cache first before being destaged to the backend disk at a later time. All reads from the array get staged into cache before being sent to the host. Even making a clone of a volume within a single array requires the blocks of data to be read from the backend, into cache, and then copied out to their new locations on the same backend. This all means that cache is extremely important.

Cache is also crucial to flash-based storage arrays, but they usually have less of it, as the backend is based on solid-state media so there isn't such a desperate need for a performance boost from cache. Flash-based storage arrays tend to use DRAM cache more for metadata caching and less for user data caching.

Performance Benefits of Cache

The raison d'etre of cache in a spinning-disk-based storage array is to boost performance. If an I/O can be satisfied from cache, without having to go to the disk on the backend, that I/O will be serviced *hundreds of times more quickly*! Basically, a well-implemented cache hides the lower performance of mechanical disks behind it.

In relation to all-flash arrays, DRAM cache is still faster than flash and can be used in a similar way, only the straight performance benefits are not as apparent.

It is necessary for write I/O to come into a storage array and be protected in two separate cache areas before an acknowledgment (ACK) can be issued to the host. That write I/O is then destaged to the backend at a later point in time. This behavior significantly improves the speed of ACKs. This modus operandi is referred to as *write-back caching*. If there are faults with cache, to the extent that incoming writes cannot be mirrored to cache (written to two separate cache areas), the array will not issue an ACK until the data is secured to the backend. This mode of operation is referred to as *write-through mode* and has a massively negative impact on the array's write performance, reducing it to the performance of the disks.

While on the topic of performance and write-through mode, it is possible to overrun the cache of a storage array, especially if the backend doesn't have enough performance to allow data in cache to be destaged fast enough during high bursts of write activity. In these scenarios, cache can fill up to a point generally referred to as the *high write pending watermark*. Once you hit this watermark, arrays tend to go into *forced flush mode*, or *emergency cache destage mode*, where they effectively operate in cache write-through mode and issue commands to hosts, forcing them to reduce the rate at which they are sending I/O. This is not a good situation to be in!

Read Cache and Write Cache

Cache memory for user data is also divided into read and write areas. Arrays vary in whether or not they allow/require you to manually divide cache resources into read cache and write cache. People tend to get religious about which approach is best. Let the user decide or let the array decide. I prefer the approach of letting the array decide, as the array can react more quickly than I can, as well as make proactive decisions on the fly, based on current I/O workloads. I do, however, see the advantage of being able to slice up my cache resources on smaller arrays with specific, well-known workloads. However, on bigger arrays with random and frequently changing workloads, I would rather leave this up to the array.

NOTE

On the topic of mirroring cache: it is common practice to mirror only *writes* to cache. Read cache doesn't need mirroring, as the data already exists on protected disk on the backend, and if the cache is lost, read data can be fetched from the backend again. If the read cache is mirrored, it unnecessarily wastes precious cache resources.

Data Cache and Control Cache

High-end storage arrays also tend to have dedicated cache areas for control data (metadata) and dedicated areas for user data. This bumps up cost but allows for higher performance and more-predictable cache performance. Disk-based storage arrays are addicted to cache for user data performance enhancement as well as caching metadata that has to be accessed quickly. All-flash arrays are less reliant on cache for user data performance enhancements but still utilize it extensively for metadata caching.

Cache Hits and Cache Misses

A *cache hit* occurs for a read request when the data being requested is already available in cache. This is sometimes called a *read hit*. Read hits are the fastest types of read. If the data does not exist in cache and has to be fetched from the backend disk, then a *cache miss*, or *read miss*, is said to have occurred. And read misses can be shockingly slower than read hits. Whether or not you get a lot of read hits is highly dependent on the I/O workload. If your workload has high referential locality, you should get good read hit results.

NOTE

Referential locality refers to how widely spread over your address space your data is. A workload with good referential locality will frequently access data that is referentially close, or covers only a small area of the address space. An example is a large database that predominantly accesses only data from the last 24 hours.

For write operations, if cache is operating correctly and an ACK can be issued to a host when data is protected in cache, this is said to be a cache hit. On a fully functioning array, you should be seeing 100 percent cache write hit stats.

Protecting Cache

As cache is so fundamental to the smooth operation of an array, it is always protected—usually via mirroring and batteries. In fact, I can't remember seeing an array pathetic enough not to have protected/mirrored cache.

There are various ways of protecting cache, and mirroring is the most common. There is more than one way to mirror cache. Some ways are better than others. At a high level, top-end arrays will take a write that hits a front-end port and duplex it to two separate cache areas in a single operation over a fast internal bus. This is fast. Lower-end systems will often take a write and commit it to cache in one controller, and then, via a second operation, copy the write data over an external bus, maybe Ethernet, to the cache in another controller. This is slower, as it involves more operations and a slower external bus.

As well as mirroring, all good storage arrays will have batteries to provide power to the cache in the event that the main electricity goes out. Remember that DRAM cache is volatile and loses its contents when you take away the power. These batteries are often used to either provide enough power to allow the array to destage the contents of cache to the backend before gracefully powering down, or to provide enough charge to the cache Dual Inline Memory Modules (DIMMs) to keep their contents secure until the power is restored.

As I've mentioned mirroring of cache, it is worth noting that only writes should be mirrored to cache. This is because read data already exists on the backend in protected nonvolatile form, so mirroring reads into cache would be wasteful of cache resources.

NOTE Protecting cache by using batteries is said to make the cache *nonvolatile*. Nonvolatile cache is often referred to as nonvolatile random access memory (NVRAM).

Cache Persistence

All good high-end storage arrays are designed in a way that cache DIMM failures, or even controller failures, do not require the array to go into cache write-through mode. This is usually achieved by implementing a grid architecture with more than two controllers. For example, a four-node array can lose up to two controller nodes before no longer being able to mirror/protect data in cache. Let's assume this four-controller node array has 24 GB of cache per node, for a total of 96 GB of cache. That cache is mirrored, meaning only 48 GB is available for write data. Now assume that a controller node dies. This reduces the available cache from 96 GB to 72 GB. As there are three surviving controllers, each with cache, writes coming into the array can still be protected in the cache of multiple nodes. Obviously, there is less overall available cache now that 24 GB has been lost with the failed controller, but there is still 36 GB of mirrored cache available for write caching.

This behavior is massively important, because if you take the cache out of a storage array, especially a spinning-disk-based storage array, you will find yourself in a world of pain!

Common Caching Algorithms and Techniques

While you don't need to know the ins and outs of caching algorithms to be a good storage pro, some knowledge of the fundamentals will hold you in good stead at some point in your career.

Prefetching is a common technique that arrays use when they detect sequential workloads. For example, if an array has received a read request for 2 MB of contiguous data to be read from the backend, the array will normally prefetch the next set of contiguous data into cache, based on there being a good probability that the host will ask for that. This is especially important on disk-based arrays, as the R/W heads will already be in the right position to fetch this data without having to perform expensive backend seek operations. Prefetching is far less useful on all-flash arrays, as all-flash arrays don't suffer from mechanical and positional latency as spinning disk drives do.

Most arrays also operate some form of *least recently used (LRU)* queuing algorithm for data in cache. LRUs are based on the principle of keeping the most recently accessed data in cache, whereas the data least recently accessed will drop out of cache. There is usually a bit more to it than this, but LRU queues are fundamental to most caching algorithms.

Flash Caching

It is becoming more and more popular to utilize flash memory as a form of level 2 (L2) cache in arrays. These configurations still have an L1 DRAM cache but augment it with an additional layer of flash cache.

Assume that you have an array with a DRAM cache fronting spinning disks. All read requests pull the read data into cache, where it stays until it falls out of the LRU queue. Normally, once this data falls out of the LRU queue, it is no longer in cache, and if you want to read it again, you have to go and fetch it from the disks on the backend. This is sloooow! If your cache is augmented with an L2 flash cache, when your data falls out of the L1 DRAM cache, it drops into the L2 flash cache. And since there's typically a lot more flash than DRAM, the data can stay there for a good while longer before being forgotten about and finally evicted from all levels of array cache. While L2 flash caches aren't as fast as L1 DRAM caches, they are still way faster than spinning disk and can be relatively cheap and large! They are highly recommended.

Flash cache can be implemented in various ways. The simplest and most crude implementation is to use flash Solid State Drives (SSD) on the backend and use them as an L2 cache. This has the performance disadvantage that they live on the same shared backend as the spinning disks. This places the flash memory quite far from the controllers and makes access to them subject to any backend contention and latency incurred by having to traverse the backend. Better implementations place the L2 flash cache closer to the controllers, such as PCIe-based flash that is on a PCIe lane on the same motherboard as the controllers and L1 DRAM cache.

External Flash Caches

Some arrays are now *starting* to integrate with flash devices installed in host servers. These systems tend to allow PCIe-based flash resources in a server to be used as an extension of array cache—usually read cache.

They do this by various means, but here is one method: When a server issues a read request to the array, the associated read data is stored in the array's cache. This data is also stored in the PCIe flash memory in the server so that it is closer to the host and does not have to traverse the storage network and be subject to the latency incurred by such an operation. Once in the server's PCIe flash memory, it can be aged out from the array's cache. This approach can work extremely well for servers with read-predominant workloads.

Lots of people like to know as much about array cache as they can. Great. I used to be one of them. I used to care about slot sizes, cache management algorithms, the hierarchical composition of caches, and so on. It is good for making you sound like you know what you're talking about, but it will do you precious little good in the real world! Also, most storage arrays allow you to do very little cache tuning, and that's probably a good thing.

Backend

The *backend* of a storage array refers to the drives, drive shelves, backend ports, and the CPUs that control them all.

CPUs

High-end enterprise storage arrays usually have dedicated CPUs for controlling the backend functions of the array. Sometimes these CPUs are responsible for the RAID parity calculations (XOR calcs), I/O optimization, and housekeeping tasks such as drive scrubbing. Mid- to low-end arrays often have fewer CPUs and as a result front-end and backend operations are controlled by the same CPUs.

Backend Ports and Connectivity

Backend ports and connectivity to disk drives are predominantly SAS these days. Most drives on the backend have SAS interfaces and speak the SAS protocol. Serial Advanced Technology Attachment (SATA) and FC drives used to be popular but are quickly being replaced by SAS.

99 percent of the time you shouldn't need to care about the protocol and connectivity of the backend. The one area in which FC had an advantage was if there was no space to add more disk shelves in the data center near the array controllers. FC-connected drives allowed you to have your drives several meters away from the controllers, in a different aisle in the data center. However, this was rarely implemented.

Real World Scenario

The Importance of Redundancy on the Backend

A customer needed to add capacity to one of their storage arrays but had no free data-center floor space adjacent to the rack with the array controllers in it. As their backend was FC based, they decided to install an expansion cabinet full of disk drives in another row several meters away from the controllers. This worked fine for a while, until the controllers suddenly lost connectivity to the disks in the expansion cabinet placed several meters away from the controller cabinet.

It turned out that a data-center engineer working under the raised floor had dropped a floor tile on an angle and damaged the FC cables connecting the controllers to the disk drive shelves. This took a long time to troubleshoot and fix, and for the duration of the fix time, the array was down. This could have been avoided by running the cables via diverse routes under the floor, but in this case this was not done. The array was taken out of action for several hours because of bad cabling and a minor data-center facilities accident.

Drives

Most drives these days, mechanical and flash, are either SATA or SAS based. There is the odd bit of FC still kicking around, but these drives are old-school now. SAS is increasingly popular in storage arrays, especially high-end storage arrays because they are dual-ported, whereas SATA is still popular in desktop and laptop computers.

Drives tend to be the industry standard 2.5-inch or 3.5-inch form factor, with 2.5-inch becoming more and more popular. And backend drives are always hot-swappable!

The backend, or at least the data on the backend, should always be protected, and the most common way of protecting data on the backend is RAID. Yes, RAID technology is extremely old, but it works, is well-known, and is solidly implemented. Most modern arrays employ modern forms of RAID such as network-based RAID and parallelized RAID. These tend to allow for better performance and faster recovery when components fail.

Most modern arrays also perform drive pooling on the backend, often referred to as *wide striping*. We cover this in more detail later in the chapter.

Power

Just about every storage array on the market will come with dual hot-swap power supplies. However, high-end arrays take things a bit further. Generally speaking, high-end storage arrays will take multiple power feeds that should be from separate sources, and tend to prefer UPS-fed three-phase power. They also sport large batteries that can be used to power the array if there is loss of power from both feeds. These batteries power the array for long enough to destage cache contents to disk or flash, or alternatively persist the contents of cache for long periods of time (one or two days is common).

Storage Array Intelligence

Yes, the term *intelligence* is a bit of a stretch, but some of this is actually starting to be clever stuff, especially the stuff that integrates with the higher layers in the stack such as operating systems, hypervisors, and applications.

Replication

In this book, *replication* refers specifically to remote replication. The raison d'etre of replication is to create remote copies (replicas) of production data. Say you have a mission-critical production database in your New York data center that you wish to replicate to your Boston data center so you have an up-to-date copy of your data in Boston in case your production systems go down in New York. Replication is what you need for this.

These copies, sometimes called *replicas*, can be used for various things, but more often than not they are used for the following:

- Providing business continuity in the event of local outages, as just described
- Test and development purposes, such as testing new models and queries against real data in an isolated environment, where they cannot impact production systems or change production data

Business Continuity

Data replication is only one component in a robust business continuity (BC) plan. It is also absolutely vital that business continuity plans are regularly tested. You do not want the first time you use your business continuity plan to be in the middle of a live disaster—hoping it will be all right on the same night. Chances are it won't!

Rehearsals are absolutely vital in managing effective business continuity plans. Test them and test them again!

WARNING

Before you look into the types of replication, it is worth pointing out that most replication technologies cannot help with recovering from logical corruption such as a virus or deleted data. This is because the corruption itself (virus or deletion) will be replicated to the target array. Backups and snapshots are what you need to recover from logical corruption!

Array-Based Replication

In *array-based replication*, the storage array controls the replication of the data. One advantage is that the heavy lifting of the replication management is handled by the storage array, so no overhead is placed on your application servers. The downside is that the storage array does not understand applications—storage array-based replication is therefore not intelligent. This means that the replicated data at the remote site may not be in an ideal state for a clean and reliable application recovery and that more data may be sent across the replication link than if application-based replication were used. Obviously, this is extremely important. For this reason, array-based replication is losing ground to the increasingly popular application-based replication.

Array-based replication almost always requires the same kind of array at both ends of the link. For example, you couldn't replicate between an EMC VMAX array in Boston and a NetApp FAS in New York.

Application-Based Replication

Application-based replication does not use storage array-based replication technologies. First, this means that the overhead of replicating data is levied on the application servers. This may not be ideal, but most systems these days have enough CPU power and other resources to be able to cope with this.

The second, and far more important, fact about application-based replication is that it is application aware, or intelligent. This means it understands the application, replicates data in a way that the application likes, and ensures that the state of the replicated copy of data is in an ideal state for the application to use for a quick and smooth recovery.

Popular examples include technologies such as Oracle Data Guard and the native replication technologies that come with Microsoft technologies such as Microsoft SQL Server and Microsoft Exchange Server.

Host-Based and Hypervisor-Based Replication

Some logical volume managers and hypervisors are starting to offer better replication technologies, and they come in various forms. Some forms of host/hypervisor-based replication technologies are neither application nor storage array aware. They simply provide the same unintelligent replication offered by storage arrays but without offloading the burden of replication to the storage array. This is not the best scenario, but it is often cheap.

There are also hypervisor-based technologies that either perform replication or plug into storage array replication technologies. An example of hypervisor-based replication technologies is VMware vSphere Replication, which is managed by VMware Site Recovery Manager.

SRM is also storage-array-based replication aware and can be integrated with array-based replication technologies. It allows the array to perform the heavy lifting work of replication but integrates this with the ability of SRM to plan and manage site failovers.

Of the two options with VMware SRM, the built-in vSphere replication is a new technology and is potentially more suited to smaller environments, whereas the option to integrate SRM with array-based replication technologies is considered more robust and scalable.

These types of replication technology are becoming increasingly popular. Beware, though. They can sometimes require both storage array replication licenses and hypervisor software licenses!

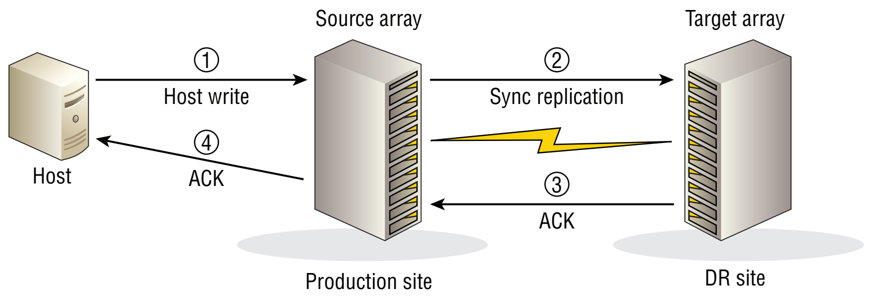
Synchronous Replication

Synchronous replication guarantees zero data loss. Great! I'm sure we all want that. However, there's always small print, and this time the small print says that synchronous replication comes with a performance cost.

Synchronous replication technologies guarantee zero data loss by making sure writes are secured at the source and target arrays before an ACK is issued to the application and the application considers the write committed. That is simple. Sadly, waiting for the write to complete at the target array can often incur a significant delay. The exact amount of delay depends on a few things, but it usually boils down to the network round-trip time (RTT) between the source and target arrays. Network RTT is also generally proportional to the distance between the source and target arrays.

Figure 3.7 shows a storyboard of array-based synchronous replication.

FIGURE 3.7 Synchronous replication storyboard



NOTE Because synchronous replication offers zero data loss, this gives a recovery point objective (RPO) of zero. At a high level, RPO is the amount of data that can be lost, measured in time units. For example, an RPO of 10 minutes means that the data used to recover an application or service can be a maximum of 10 minutes out-of-date at the time the data loss occurred. Therefore, RPOs dictate the business continuity technologies used to protect business data. For example, a daily backup of an application is no use if the RPO for that application is 1 hour.

Replication Distances and Latency

As a very loose ballpark figure, a distance of 75 miles between source and target arrays may incur an RTT of 1–2 milliseconds (ms). That is 1–2 ms of latency that would not be incurred if you were not synchronously replicating your data.

While on the topic of distances, a good ballpark figure for maximum distance you might want to cover in a synchronous replication configuration might be ~100 miles. However, it is important that you consult your array vendor over things like this, as they may impose other limitations and offer recommendations depending on their specific technology. This is not something you want to design on your own and get wrong.

Make sure that you include the array vendor as well as the network team when dealing with site-to-site replication links.

NOTE Always make sure that when referring to the distance between data centers, you are indicating the route taken by the network and not the distance if travelling by car. The distance could be 60 miles by car but a lot longer via the route the network takes.

NOTE It is also important to be able to test and guarantee the latency of the link. You may want to get latency guarantees from your Wide Area Network (WAN) provider.

Replication Link Considerations

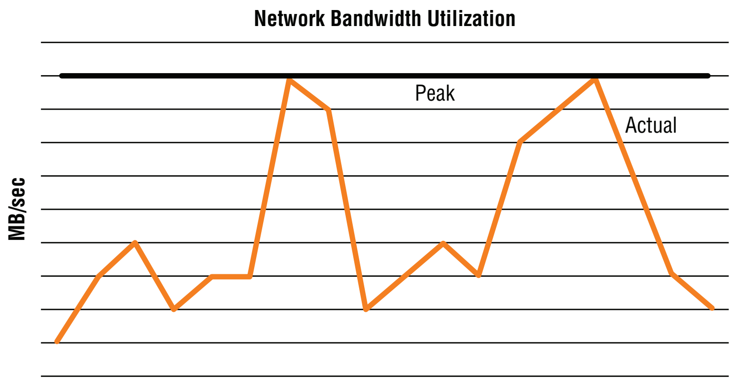
When deploying synchronous replication, you should also be savvy to the importance of the network links connecting your source and target arrays. A flaky link that is constantly up and down will mean that your solution cannot guarantee zero data loss without stopping writes to the application while the network link is down. Most organizations will allow writes to the application to continue while replication is down; however, for the duration of the replication outage, the remote replica will be out of sync. If you lose the primary copy while replication is down, your replica at the remote site will not be up-to-date, your Service Level Agreement (SLA) will be broken, and you will be facing a data-loss scenario if you need to bring the applications up in DR.

NOTE Although rare, some organizations do have applications that they fence off (stop writing to) if the remote replica cannot be kept in sync. These tend to be special-case applications.

Because of all of this, it is important to have multiple, reliable, diverse replication links between your sites. All tier 1 data centers will be twin tailed—two links between sites that each take diverse routes.

You will also want to consider the sizing of your replication link. If you do not size your link to be able to cope with your largest bursts of traffic, the link will be saturated when you experience high bursts of write data, and subsequently the performance of your applications will drop. Some organizations are happy to take this on the chin, as it means they don't have to oversize their replication links to deal with peak traffic. The important thing is that you know your requirements when sizing and speccing replication links. Figure 3.8 shows network bandwidth utilization at hourly intervals. In order for this solution to ensure that the remote replica does not lag behind the production source volume, the network link will need to be sized to cope with peak traffic.

FIGURE 3.8 Synchronous remote replication bandwidth requirements



Asynchronous Replication

The major difference between synchronous and asynchronous replication is that with asynchronous replication, each write is considered complete when the local array acknowledges it. There is no wait for the write to be committed to the replica volume before issuing the ACK.

This means several things.

First, asynchronous does not provide zero data loss. In fact, asynchronous replication guarantees to lose data for you! How much data is lost depends entirely on your configuration. For example, if your replication solution replicates only every 5 minutes, you could lose slightly more than 5 minutes' worth of data.

Next, and somewhat more positively, asynchronous replication does not incur the performance penalty that synchronous replication does. This is because the write to the remote array is done in a lazy fashion at a later time.

It also means that the distances between your source and target array can be much larger than in synchronous configurations. This is because the RTT between the source and target array no longer matters. You can theoretically place your source and target arrays at opposite ends of the planet.

There is also no need to spec your network connections between sites to cater for peak demand. However, you should still spec your network link so that you don't slip outside of your agreed RPO.

All of this is good, assuming you can sleep at night knowing you will lose some data if you ever have to invoke DR.

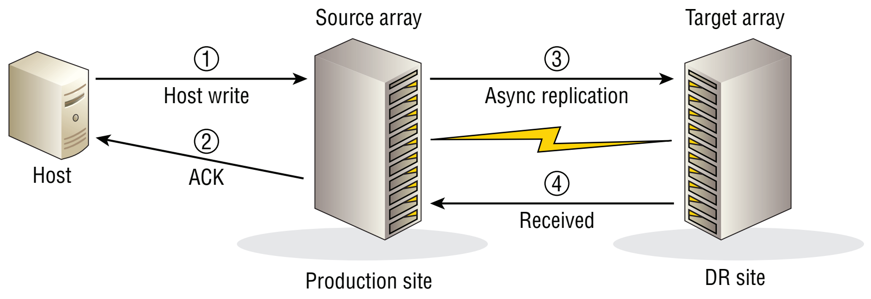
NOTE

Invoking DR (disaster recovery) refers to bringing an application up by using its replicated storage. This usually involves failing the application—including the server, network, and storage—over to the standby server, network, and storage in the remote disaster recovery data center.

Storage arrays tend to implement asynchronous replication, shown in Figure 3.9, in one of two ways:

- Snapshot based
- Journal based

FIGURE 3.9 Asynchronous remote replication



The difference to note between Figure 3.9 and Figure 3.7 is that the host receives an ACK—and can therefore continue issuing more I/O—after step 2 in Figure 3.9.

Snapshot-Based Asynchronous Replication

In snapshot-based asynchronous replication, the source array takes periodic *point-in-time snapshots* of the source volume and then replicates the snapshot data over the wire to the target array, where it is applied to the replica volume.

This type of replication is schedule based, meaning that you schedule the snaps and replication interval according to your required recovery point objective (RPO). Assume you have agreed to an SLA with the business to be able to recover data to within 10 minutes of the time at which an incident occurred. Configuring a replication interval of 5 minutes will meet that RPO and ensure that the replica volumes never lag more than ~5 minutes behind the source volumes.

Good storage arrays have an interface that allows you to specify your RPO, and it will configure the replication specifics for you to ensure that your RPO SLA is met. Better storage arrays are RPO driven to even greater depths. For example, if you have some volumes configured for a 15-minute RPO and others for a 20-minute RPO, if the array encounters replication congestion, the array will prioritize replication based on which RPOs are closest to breaking. That borders on array intelligence!

If your storage array is stuck in the '90s and doesn't understand RPOs and SLAs, you will have to manually consider these things when configuring replication.

WARNING

Be careful. An RPO of 10 minutes *does not* mean you can configure a replication update interval of 10 minutes! This is essentially because your snapshot data won't instantly arrive at the target array the moment it is sent. It takes time for data to be transmitted across a network, and often the wire used for asynchronous replication is a relatively cheap, low-bandwidth wire. Assuming it take 2 minutes for all your data to arrive at the target array, that is potentially a 12-minute RPO, not a 10-minute one.

Usually, if your array supports snapshot-based asynchronous replication, that array will use the same snapshot engine that it uses to take local snapshots. Depending on your array, this can mean that replication-based snapshots eat into the maximum number of snapshots supported on the array. So if your array supports a maximum of 1,024 snapshots, and you are replicating 256 volumes by using snapshot-based asynchronous replication, you have probably eaten 256 of the array's maximum 1,024 snaps.

WARNING

With snapshot-based asynchronous replication, beware of snapshot extent size—the granularity at which snapshots are grown. If the snapshot extent size is 64 KB, but all that you update between replication intervals is a single 4 KB block, what will actually be replicated over the wire is 64 KB—a single snapshot extent. No big deal, right? Now assume several large arrays sharing a remote replication link, each using snapshot-based replication. You update 1,000 unique 4 KB blocks and assume that this will replicate just under 4 MB of data across the wire. However, each of these 4 KB updates was in a unique extent on each volume, meaning that you will end up replicating 1,000 × 64 KB instead of 1,000 × 4 KB. The difference is then pretty large—62.5 MB instead of 3.9 MB!

A good thing about snapshot-based replication is that as long as your array has decent snapshot technology, it will coalesce writes. This means that if your app has updated the same data block 1,000 times since the last replication interval, only the most recent update of that data block will be sent across the wire with the next set of deltas, rather than all 1,000 updates.

Journal-Based Asynchronous Replication

Journal-based asynchronous replication technologies buffer write data to dedicated *journal volumes*, sometimes referred to as *write intent logs*. They have to be sized appropriately (usually oversized) in order to cope with large traffic bursts or times when the replication links are down. If you size these journal volumes too small, replication will break during large bursts or extended periods of the replication link downtime. On the other hand, if you size them too big, you're wasting space 99 percent of the time. It can be considered a balancing act or a dark art. Either way, it is quite complex from a design perspective.

When write I/Os come into the source array, they hit cache as usual, and the array issues the ACK to the host. The write I/O is also tagged with metadata to indicate that it is destined for a replicated volume, to ensure that the data is also asynchronously copied to the local journal volumes. The data is then destaged to disk as per standard cache destaging. At or around this time, the data is also written to the journal volume. From there, it is replicated to the target array at a time determined by the specifics of the array's replication technology. Generally speaking, though, journal-based replication does not lag very far behind the source volume.

Good arrays will apply write-sequencing metadata to the data buffered to the journal volumes so that proper write ordering is maintained when data is written to the target volume on the target array. Once the data is committed at the target array, it can be released from the journal volumes on the source array.

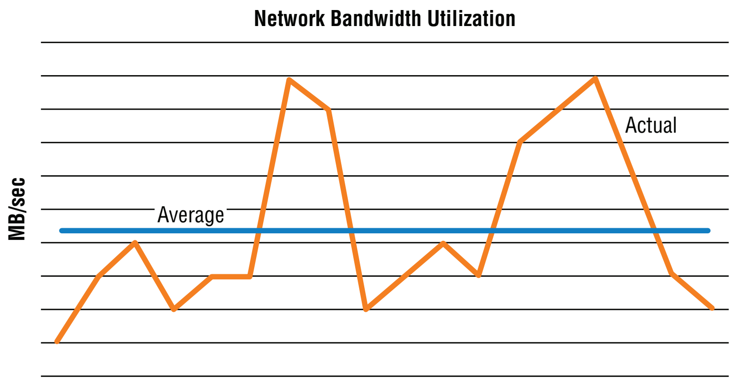
Implementations differ as to whether the source array pushes updates to the target array or whether the target array pulls the updates. For your day-to-day job, things like this should not concern you, certainly not as much as making sure you meet your SLAs and RPOs.

WARNING

Although asynchronous replication doesn't require top-notch network bandwidth between source and target arrays, make sure your network connections aren't the runt of the network litter, relegated to feeding on the scraps left over after everyone else has had their share. If you don't give the links enough bandwidth, you will be operating outside your SLAs too often, risking not only your job but your company's data.

Figure 3.10 shows actual bandwidth utilization captured at hourly intervals, with an average line included. Asynchronous replication links do not need to be sized for peak requirements.

FIGURE 3.10 Bandwidth requirements for asynchronous replication



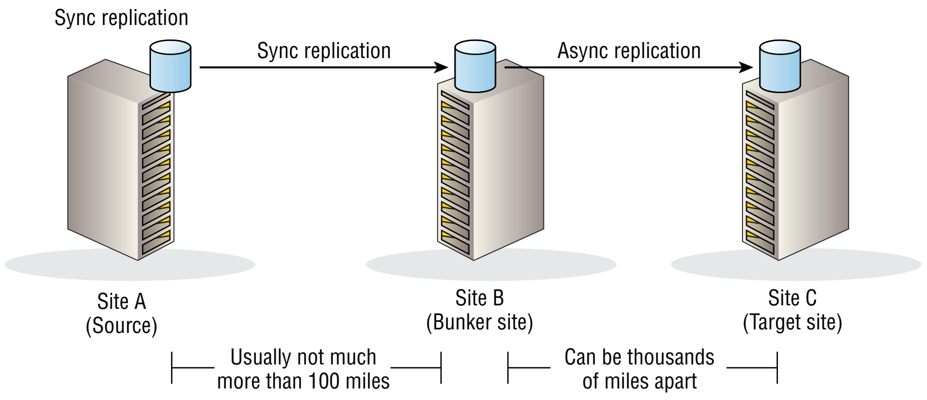
Replication Topologies

Depending on your array technology, various replication topologies are available. These can include more than two sites and utilize both synchronous and asynchronous replication technologies. It tends to be the higher-end, enterprise-class arrays that support the most replication topologies. The midrange arrays tend to support only more-basic topologies.

Three-Site Cascade

The *three-site cascade* replication topology utilizes an intermediate site, sometimes referred to as a *bunker site*, to connect the source array and the target array. Figure 3.11 shows an example of this topology.

FIGURE 3.11 Three-site cascade topology



As shown in Figure 3.11, three-site cascade utilizes both synchronous and asynchronous replication—synchronous replication from the source site to the bunker site, and then asynchronous from the bunker site to the target site.

A major use case for three-site cascade topologies is to enable application recovery with zero data loss in the event of a localized issue in your primary data center that requires you to invoke disaster recovery plans. In this situation, you can bring your applications up in the bunker site with potentially zero data loss. However, if a major disaster renders both the primary data center and the bunker site inoperable, you have the third copy at the remote site. Because the replication between the bunker site and the target site is asynchronous, the distance can be large enough that even a major local disaster that affects the source and bunker sites will not affect the target site. And if a disaster big enough to affect the target site does occur, there is a good chance that recovering your company's applications will be the last thing on your mind!

Three-site cascade also allows your applications to remain protected even when the source site is down. This is because there are still two sites with replication links between them, and replication between these two sites can still operate.

The major weakness of the three-site cascade model is that the failure of the bunker site also affects the target site. Basically, if you lose the bunker site, your target site starts lagging further and further behind production, and your RPO becomes dangerously large.

This model is also used with the secondary array in the primary data center, rather than a remote bunker site, to protect against a failure of the primary array rather than a full site failure.

Three-Site Multitarget

The *three-site multitarget* topology has the source array simultaneously replicate to two target arrays. One target array is at a relatively close bunker site—close enough for synchronous replication. The other target is a long distance away and is replicated to asynchronously. This configuration is shown in Figure 3.12.

The major advantage that three-site multitarget has over three-site cascade is that the failure of the bunker site has no effect on replication to the target site. In this respect, it is more robust.

The major weakness of the three-site multitarget topology is that if the source site is lost, no replication exists between the bunker site and the target site. It can also place a heavier load on the primary array.

Three-Site Triangle

The *three-site triangle* topology is similar to three-site multitarget and offers all the same options, plus the addition of a standby replication link between the bunker and target site. Under normal operating circumstances, this additional link is not actively sending replication traffic, but can be enabled if the source site becomes unavailable. The three-site triangle topology is shown in Figure 3.13.

FIGURE 3.12 Three-site multitarget

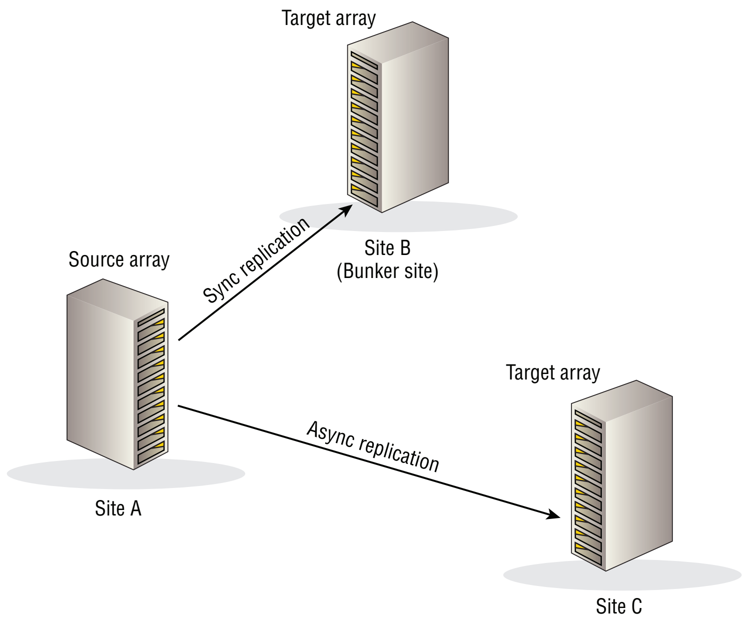
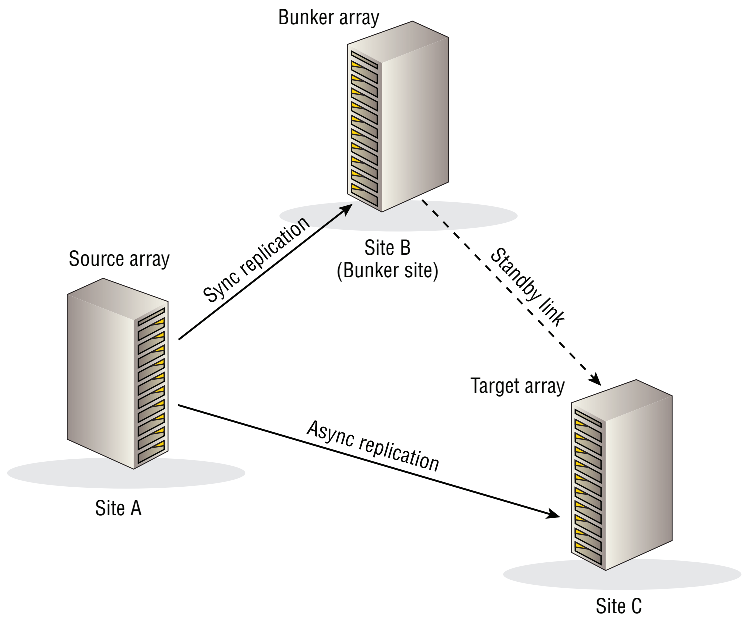


FIGURE 3.13 Three-site triangle replication topology



If the source site is lost, synchronizing replication between the bunker site and the target site does not require a full sync. The surviving arrays in the bunker and target site are able to communicate and determine which updates are missing at the target site, allowing for an incremental update from the bunker array to the target array. This allows you to get your application up and running again and have it protected by remote replication in a short period of time.

NOTE Not all storage arrays support all replication topologies. It is vital that you check with your array vendor or array documentation before assuming that your array supports a particular multisite replication topology.

Local Snapshots

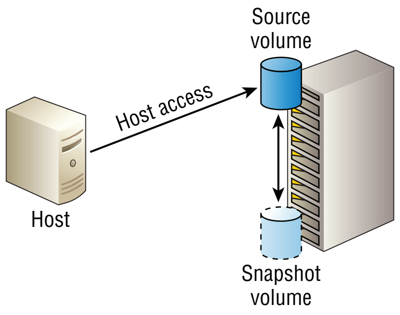
Snapshots are nothing new, and in some cases neither is the technology behind them. Some snapshot implementations are ancient. We'll uncover some of these.

First, let's agree on our terminology: when referring to *snapshots*, I'm talking about local copies of data. And by *local*, I mean the snapshot exists on the same array as the source volume. Generally speaking, snapshots are point-in-time (PIT) copies.

These two facts—local and point-in-time—are the two major differences between local snapshots and remote replicas. Local snapshots are copies of production volumes created and maintained on the local array, whereas remote replicas are kept on a remote array. Also, local snapshots are point-in-time copies, whereas remote replicas are usually kept in sync, or semi-sync, with the primary production volume.

Finally, I use the terms *source volume* and *primary volume* to refer to the live production volume. And I use the term *snapshot* to refer to the point-in-time copy of the live production volume. This is shown in Figure 3.14.

FIGURE 3.14 Simple snapshot example



Snapshots and clones are created instantaneously, and depending on your array, snapshots and clones can be marked as read-only or read/write. All good arrays support read-only and read-write snapshots.

WARNING

Snapshots are not backups! Take this statement, burn this into your brain, RAID protect it with at least double parity, and pin it to the high-performance cache of your brain so that it is always there for quick recall. Snapshots are not backups! Why? If you lose the RAID set, the array, or the site that hosts the primary volumes, you lose the snapshots as well. You do not want to be in this predicament.

Array-Based Snapshots

Two types of array-based snapshot technologies are commonly in use:

- Space-efficient snapshots
- Full clones

While they may go by various names, this book sticks to these two names. This section covers topics such as snapshot extent size, space reservation, the number of snapshots, how tiering can be used with snapshots, snapshot consistency, and the difference between copy-on-write and redirect-on-write snapshots.

Space-Efficient Snapshots

Space-efficient snapshots are generally pointer based. That tells you two important things:

Space efficient means that the snapshot contains only the parts of the primary volume that have changed since the snapshot was taken.

Pointer based means that none of the data that hasn't changed since the time of snapshot creation is copied to the snapshot. Instead, pointers point back to the data in the original volume for any data that hasn't changed since the time of snapshot creation.

Let's look at a quick and simple example with a figure or two.

Assume that you have a primary volume called `LegendaryVol`. `LegendaryVol` has 100 blocks numbered 0–99. At 1 p.m. on Monday, you make a space-efficient snapshot of `LegendaryVol`, at which point in time, all blocks in `LegendaryVol` are set to binary zero (0). Your snapshot is called `LegendaryVol.snap`. At the time you create the snapshot, `LegendaryVol.snap` consumes *no* space. If a host were to mount this snapshot and read its contents, all read I/O would be redirected back to the original data contained in `LegendaryVol`. All that `LegendaryVol.snap` consumes is pointers in memory that redirect read I/O back to the primary volume where the data exists.

Now let's assume that by 2 p.m. `LegendaryVol` has had blocks 0–9 updated to be binary ones (1). Any time a host wants to read data in `LegendaryVol.snap`, it will be redirected back to the source volume (`LegendaryVol`) *except for reading blocks 0–9*. This is because blocks 0–9 have changed in the primary volume, and in order to make sure that the snapshot looks exactly like the primary volume did at the point in time when the snap was taken (1 p.m.), the original contents were also copied to the snapshot.

So, at 2 p.m., the snapshot is still an exact image for how the primary volume was at 1 p.m.—all blocks are zeros. Job done. Also, the snapshot is space efficient because all unchanged data since the time the snapshot was created is still just pointers back to the original data, and the snapshot consumes only the space required to preserve the data that has changed since.

Full-Clone Snapshots

Full clones are not space efficient and therefore are not pointer based. They are, however, point-in-time copies of primary volumes.

At the time a full clone is created, an exact full copy of the primary volume is taken. This means that if the primary volume consumes 2 TB of storage at the time the clone was taken, each and every full clone of that volume will also consume 2 TB. This obviously means that full clones can be major consumers of disk space.

So why bother with full clones rather than space-efficient snapshots? There are a couple of common reasons:

- Reading and writing to a full clone has no direct impact on the primary volume. This is because the full clone is fully independent of the primary volume and contains no pointers. So you can hammer the clone with read and write requests, and this will have no impact on the performance of the primary volume. Snapshots, on the other hand, share a lot of data with their primary volumes, meaning that if you hammer the snapshot with intensive read and write I/O, you will be impacting the performance of the primary volume as well.
- Full clones are fully independent of the primary volumes from which they were created. This means that if the physical drives that the primary volume is on fail, the clone will not be impacted (assuming you don't make the mistake of keeping the primary and full clone on the same drives). In contrast, with snapshots, if the disks behind the primary volume fail, the snapshot will also be affected because of the way the snapshot just points back to the primary volume for much of its content.

In the modern IT world, where budgets are tight and ever shrinking, snapshots tend to be far more popular than full clones, so we'll spend a bit more time covering space-efficient snapshots.

Snapshot Extent Size

When it comes to space-efficient snapshots, size definitely matters. And bigger is not better!

If your array has a snapshot *extent size* (sometimes referred to as *granularity*) of 4 KB, it will be far more space efficient than an array with a snapshot granularity of 128 KB. Here is a quick example. Assume you make 100 changes to your primary volume since you created a snapshot. If your snapshot extent size is 4 KB, this will amount to a maximum of 400 KB consumed by your snapshot. On the other hand, if your snapshot extent size is 128 KB, the maximum space consumed by your snapshot just jumped up to 12,800 KB!

As another example, if you make a 4 KB update to a primary volume that has a snapshot, and the snapshot extent size is 128 KB, an additional 128 KB may have to be allocated to your snapshot. This is because the *extent* is the minimum unit of snapshot growth.

While this might not sound like the end of the world, consider the impact this has when you have lots of snaps and you are keeping them for long periods of time. This space inefficiency soon adds up. This also has an impact on how much data is sent over the wire in asynchronous replication configurations that are based on snapshots. So when it comes to snapshot granularity or extent size, smaller is definitely better.

Reserving Snapshot Space

On arrays based on legacy architectures, you are required to guess how much space your snapshots *might* consume, and then reserve that space up front! That's right, on day one you have to open up your wallet, and purchase and set aside space for potential snapshot use, space that might never be used. This type of architecture is made even worse by the fact that in many instances you have to prepare for the worst case and often end up ring-fencing far too much space. Legacy architectures like this are inexcusable in the modern IT world. Unfortunately, legacy array architectures that are built on this approach still exist and are still being sold! Beware.

A more modern approach is to have your snapshots dynamically allocate themselves space from a shared pool (not usually the same pool as the primary volume). This shared pool will be used by other volumes—primary volumes and snapshot volumes—and you will be allowed to set policies to deal with situations where the amount of space in the pool is running short. For example, you might set a policy that deletes the oldest snapshots when the pool hits 90 percent full to ensure that thinly provisioned primary volumes do not suffer at the expense of snapshots.

Too Many Snapshots

On most arrays, snapshots are considered volumes. This means that if your array supports only 8,192 volumes, the total number of primary volumes and snapshot volumes combined can be 8,192. It does not usually mean 8,192 volumes plus snapshots. Always check with your array documentation!

Real World Scenario

What Can Happen If You Don't Understand Your Array Limits

A company was improving its Exchange backup-and-recovery solution by using array-based snapshots integrated with the Microsoft Volume Shadow Copy Service (VSS). The grand plan was to take a single snapshot of each Exchange Mailbox Server each day, and keep these snapshots for 30 days. They had a dedicated block storage array for Exchange that supported 1,024 volumes (including snapshots). What they didn't consider at the outset was that although they didn't have many Exchange servers, each Exchange server had multiple volumes presented to it. In total, they had 41 volumes that needed snapshotting. They obviously hit a brick wall at around day 23 or 24, when the number of primary volumes and snapshots on the array hit 1,024. Suddenly the array could no longer create any more snapshots or volumes.

To resolve this, the solution had to be re-architected to provide 21 days' worth of snapshots rather than 30.

We should also point out that while this company was using snapshots for Exchange backups, the snapshots were being employed to augment existing backup processes. The array-based snapshot technology was being used to speed up and smooth out the process of taking an application-consistent copy of the Exchange environment. These snaps were then being copied to tape, and the tapes were taken offsite. The snapshot technology was being used to improve the reliability and performance of the backups as well as provide a potentially faster recovery method. They could still fall back to tapes for recovery if they lost their snaps.

Tiering Snapshots

Most modern arrays also allow snapshot volumes to be controlled by their auto-tiering algorithms, thus enabling older snapshots to trickle down through the tiers and end their lives consuming slow, cheap storage. For example, assume your array is set up and working so that data blocks that have not been accessed for more than 10 days are automatically tiered down to a lower-tier disk. Any snapshot that has not been accessed for 10 days or more will stop consuming high-performance disk. This allows your snapshots to consume high-performance disk early in its life when it is more likely to be used, but then put its feet up on a lower-tier disk as it prepares for retirement.

Crash-Consistent and Application-Aware Snapshots

Similar to array-based replication, array-based snapshots in their vanilla form have no intelligence—they are not application aware! Snapshots that are not application aware are considered *crash consistent*. This oxymoron of a term basically means that the snapshots could be worthless as a vehicle of application recovery; in the eyes of the application, the data contained in the snapshot is only as good as if the server and application had actually crashed, and might be considered corrupt. This is because most applications and databases cache data in local buffers, and lazily commit these writes to disk. This tends to be for performance reasons. This means that at any point in time, the data on disk does not accurately represent the state of the application.

Fortunately, array-based snapshots can be made application aware! A common example is the VSS. Most good arrays can integrate with Microsoft VSS so that the array and application coordinate the snapshot process, ensuring that the resulting snapshots are *application-consistent snapshots*—snapshots that the application can be effectively and efficiently recovered from. When snapshotting or cloning applications in the Microsoft world, this is most definitely what you want!

However, if your application does not support VSS or if you are running your application on Linux, you will probably be able to run scripts that place your application into hot backup mode while consistent snapshots can be taken on the array.

NOTE Microsoft VSS can be used for making application-consistent, array-based snapshots for applications such as Microsoft SQL Server and Microsoft Exchange.

Copy-on-Write Snapshots

Copy-on-write (CoW) snapshots are old-school and on the decline. On the downside, they have the overhead of first-write performance, but on the upside they maintain the contiguous layout of the primary volume. Let's look at both quickly:

- The copy-on-first-write penalty causes each write to the primary volume to be turned into three I/Os: read original data, write original data to the snapshot, and write new data to the primary volume. This can also increase write amplification on flash-based volumes, reducing the life span of the flash media.
- The good thing about CoW snapshots is that they preserve the original contiguous layout of the primary volume. This is covered when talking about redirect-on-write snapshots next.

Redirect-on-Write Snapshots

Redirect-on-write (RoW) snapshots, sometimes referred to as *allocate-on-write*, work differently than copy-on-write. When a write I/O comes into a block that is protected by a snapshot, the original data in the block is not changed. Instead it is frozen and made part of the snapshot, the updated data is simply written to a new location, and the metadata table for the primary volume is updated.

Here is an example: A volume comprises 100 contiguous blocks, 0–99. You snap the volume and then later want to update block 50. With redirect-on-write, block 50's contents will not change, but will be frozen in their original condition and made part of the address space of the snapshot volume. The write data that was destined to block 50 is written elsewhere (let's say block 365). The metadata for the primary volume is then updated so that logical block 50 now maps to physical block 365 on the disk (metadata writes are so fast that they don't impact volume performance).

This behavior starts to fragment the layout of the primary volume, resulting in it eventually becoming heavily fragmented, or noncontiguous. Such noncontiguous layouts can severely impact the performance of a volume based on spinning disks. It is not a problem, though, for volumes based on flash drives.

Redirect snapshots are increasing in popularity, especially in all-flash arrays, where the noncontiguous layout of the primary volume that results from RoW snapshots does not cause performance problems.

Application- and Host-Based Snapshots

Generally speaking, application- and host-based snapshots such as those offered by Logical Volume Manager (LVM) technologies do not offload any of the legwork to the array. They can also suffer from having to issue two writes to disk for every one write from the application, and are commonly limited to presenting the resulting snapshot only to the same host that is running the LVM. They are, however, widely used in smaller shops and development environments.

Thin Provisioning

In the storage world, *thin provisioning (TP)* is a relatively new technology, but one that has been enthusiastically accepted and widely adopted at a fairly rapid pace. The major reason behind its rapid adoption is that it helps avoid wasting space and can therefore save money. And anyone who is familiar with storage environments knows that a lot of capacity wasting occurs out there.

Before we get into the details, let's just agree on our terminology up front. This book uses the terms *thick volumes* and *thin volumes*, but be aware that other terms exist, such as *virtual volumes* and *traditional volumes*, respectively.

Thick Volumes

Let's start with a review of thick volumes. These are the simplest to understand and have been around since year zero. The theory behind thick volumes is that they reserve 100 percent of their configured capacity on day one.

Let's look at a quick example. Imagine you have a brand new storage array with 100 TB of usable storage. You create a single 4 TB thick volume on that array and do nothing with it. You don't even present it to a host. You have instantly reduced the available capacity on your array from 100 TB to 96 TB. That's fine if you plan on using that 4 TB soon. But often that isn't the case. Often that 4 TB will go hideously unused for a long time. And this is where thick volumes show their real weakness. Most people grab as much storage as they can for themselves and then never use it.

Let's look a little more closely at the problem.

Most application owners who request storage massively overestimate their requirements. You wouldn't want to get caught with too little capacity for your application, right? It's not uncommon for people to make blind guesses about their capacity requirements, and then double that guess just in case their estimate is wrong, and then double it again just to be extra safe. Take an initial finger-in-the-air estimate of 250 GB, double it to 500 GB just in case your estimate is wrong, and then double it again to 1 TB just to be extra safe. On day one, the thick volume consumes 1 TB of physical space on the arrays even though the application needs only 50 GB on day one. Then after three years of use, the application has still consumed only a measly 240 GB even though the volume is hogging 1 TB of physical capacity—effectively wasting over 750 GB.

Maybe this isn't such a problem in a flush economy or if your company owns acres of land dedicated to growing money trees. However, if you live in the real world, this amounts to gross wasting of expensive company assets and is rightly becoming a capital crime in many organizations.

Thin Volumes

This is where thin volumes come to the rescue. Whereas a 1 TB thick LUN reserves itself 1 TB of physical capacity at the time it is created, thin volumes reserve nothing up front and work similarly to files, which grow in size only as they're written to.

NOTE

Actually, thin volumes do consume space as soon as they are created. However, this space is usually so small as to be insignificant. Most arrays will reserve a few megabytes of space for data to land on, with the remainder being metadata pointers in cache memory.

Let's look again at our example 1 TB volume, only this time as a thin volume. This time our thin 1 TB volume reserves no physical capacity up front and consumes only 50 GB of physical capacity on day one—because the application has written only 50 GB of data. And after three years, it consumes only 240 GB. That leaves the other 760 GB for the array to allocate elsewhere. This is a no-brainer, right?

Thin Provisioning Extent Size

The thin-provisioning world is similar to the space-efficient snapshot world: size matters. And as with the space-efficient snapshot world, smaller is better.

TP extent size is the growth unit applied to a thin volume. Assume that your 1 TB TP volume starts its life as 0 MB. If the TP extent size is 128 MB, as soon as the host writes 1 KB to that volume, it will consume a single 128 MB TP extent on the storage array. Likewise, if your TP extent size was 42 MB, you would have consumed only 42 MB of backend space with that 1 KB write. Or if your TP extent size was 1 GB, you would consume a relatively whopping 1 GB for that 1 KB write. This should make it easy to see that from a capacity perspective, smaller extent sizes are better.

TP extents are sometimes referred to as *pages*.

Thin Provisioning and Performance

The one place in which a very small extent size might not be beneficial is on the performance front. You see, each time a new extent has to be assigned to a thin volume, there is a small (very small) performance overhead while the array allocates a new extent and adds it to the metadata map of the volume. However, this is rarely considered a real issue, and a large TP extent size is almost always a sign that the array is not powerful enough to handle and map large numbers of extents.

Generally speaking, though, because thin provisioning is built on top of wide-striping/pooling, thinly provisioned volumes are spread across most, if not all, drives on the backend, giving them access to all the IOPS and MB/sec on the entire backend. This tends to lead to better performance.

One use case where thinly provisioned volumes can yield lower performance is with heavily sequential reads in spinning-disk-based arrays. Sequential reads like data to be laid out sequentially on spinning disk in order to minimize head movement. As TP volumes consume drive space only on the backend as data is written to them, this data tends not to be from contiguous areas, leading to the address space of the volume being fragmented on the array's backend. (Host-based defragmentation tools can do nothing about this.) This can lead to TP volumes performing significantly lower than thick volumes for this particular workload type.

Overprovisioning

Thin provisioning naturally leads to the ability to overprovision an array. Let's look at an example.

Assume an array with 100 TB of physically installed usable capacity. This array has 100 attached servers (we're using nice round numbers to keep it simple). Each server has been allocated 2 TB of storage, making a total of 200 TB of allocated storage. As the array has only 100 TB of physical storage, this array is 100 percent overprovisioned.

This overprovisioning of capacity would not be possible when using thick volumes, as each thick volume needs to reserve its entire capacity at the point of creation. Once the 50th 2 TB volume was created, all the array space would be used up, and no further volumes could be created.

Overprovisioning Financial Benefits

Let's look at overprovisioning from a financial perspective. If we assume a cost of $4,000 per terabyte, our 100 TB storage array would cost $400,000. However, if we didn't overprovision this array and had to buy 200 TB to cater to the 200 TB of storage we provisioned in our previous example, the array would cost $800,000. So it's not too hard to see the financial benefits of overprovisioning.

Risks of Overprovisioning

While overprovisioning is increasingly common today, *it should be done only when you understand your capacity usage and trending as well as your company's purchasing cycle!* And every aspect of it should be thoroughly tested—including reporting and trending—before deploying in production environments.

The major risk is a bank run on your storage. If you overprovision a storage array and run out of capacity, all connected hosts might be unable to write to the array, and you should expect to lose your job. So good planning and good management of an overprovisioned array is paramount!

It should be noted that overprovisioning is something of a one-time trick. You can use it to your benefit only once. Thereafter, it takes a lot of looking after to ensure that you don't run out of storage. Assume again you have a 100 TB array and have provisioned 100 TB, but only 50 TB is actually being used. You can overprovision, to say 180 TB provisioned storage, giving you a total of 180 TB exported storage. Excellent, you have just magicked up 80 TB for a total capital expenditure cost of $0.00. Management will love you.

However, you won't be able to pull the same trick again next year. In fact, next year you will be operating with the additional risk that if you don't manage it correctly, you could run out of storage and bring all of your applications down. This increased risk is real and absolutely must not be ignored or overlooked.

Because of the inherent risks of overprovisioning, it is vital that senior management understand the risks, know what they are getting into, and formally sign off on the use of overprovisioning.

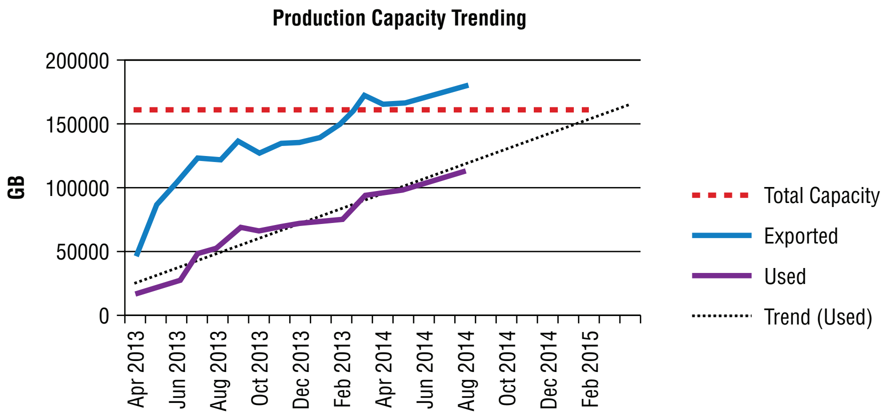
Trending Overprovisioning

After you have started overprovisioning, you need to closely monitor and trend capacity utilization. The following are key metrics to monitor in an overprovisioned environment:

- Installed capacity
- Provisioned/allocated capacity
- Used capacity

Each of these needs recording on at least a monthly basis and needs trending. Figure 3.15 shows values of these three metrics over a 16-month period to date. It also has a trend line projecting future growth of used capacity. The trend line shows that based on current growth rates, this array will run out of space around January or February 2015.

FIGURE 3.15 Overprovisioning chart



When to Purchase Capacity on Overprovisioned Arrays

You need to establish agreed-upon thresholds with senior management that trigger new capacity purchases, and it is imperative that these are decided with an understanding of your company's purchasing process. For example, if your company takes a long time to approve purchases, and your vendor takes a long time to quote and ship storage, you will not want to run things too close to the bone.

Many companies use at least two trigger points for capacity purchases. Once the first trigger point is reached, a purchase order for more capacity is raised. Some common trigger points include the following:

- Actual available capacity (installed capacity minus used capacity)
- Percentage of available capacity (percentage of installed capacity that is unused)
- Overprovisioning percentage

Based on these triggers, you might implement the following trigger rules:

- As soon as free capacity (installed capacity minus used capacity) reaches less than 10 TB, you kick off the purchasing cycle.
- Once the percentage of available capacity (installed minus used, expressed as a percentage) reaches less than 10 percent, you might kick off the purchasing cycle.
- Once the array becomes more than 60 percent overprovisioned, you might kick off the capacity purchasing cycle.

Each of these rules depends heavily on your environment, and you must understand the growth characteristics of your environment. For example, waiting until an array has only 10 TB of free capacity will be of no use to you if your growth rate on that array is 15 TB per month and it takes you an average of eight weeks to land additional capacity in your environment. In that case, a value of 50 TB or more free capacity might be a better value.

Space Reclamation

Space reclamation is all about keeping thin volumes thin. Thin-provisioning volumes allow you to avoid preallocating capacity up front and allocate only on demand. However, your thin volumes will quickly become bloated if your thin-provisioning array is unable to recognize when the host deletes data. This is where space reclamation comes into play.

Zero Space Reclamation

At a high level, zero space reclamation is the act of recognizing deleted space in a thin volume and releasing that space back to the free pool so that it can be used by other volumes. However, the secret sauce lies in the array being able to know when space is no longer used by a host.

Space Reclamation the Old-School Way

Traditionally, when an OS/hypervisor/filesystem deleted data, it didn't actually delete the data. Instead, it just marked the deleted data as no longer required. That wasn't much use to arrays, as they had no knowledge that this data was deleted, so they were unable to release the extents from the TP volume. In these cases, scripts or tools that would physically write binary zeros to the deleted areas of data were required in order for the array to know that there was capacity that could be released.

NOTE Zero space reclamation works on the principle of binary zeros. For an array to reclaim unused space, that space must be *zeroed out*—the act of writing zeros to the deleted regions.

Host-Integration Space Reclamation

Modern operating systems such as Red Hat Enterprise Linux 6 (RHEL), Windows Server 2012, and vSphere 5 know all about thin provisioning and the importance of keeping thin volumes thin, so they are designed to work smoothly with array-based zero-space-reclamation technologies.

To keep thin volumes thin, these operating systems implement the T10 standards-based `UNMAP` command that is designed specifically for informing storage arrays of regions that can be reclaimed. The `UNMAP` command informs the array that certain Logical Block Address (LBA) ranges are no longer in use, and the array can then release them from the TP volume and assign them back to the free pool. There is no more need for scripts and tools to zero out unused space; the OS and filesystem now take care of this for you.

Inline or Post-process Space Reclamation

Some arrays are clever enough to recognize incoming zeros on the fly and dynamically release TP extents as streams of zeros come into the array. Other arrays are not that clever and require an administrator to manually run post-process space-reclamation jobs in order to free up space that has been zeroed out on the host.

Both approaches work, but the inline approach is obviously a lot simpler and better overall, as long as it can be implemented in a way that will not impact array performance.

Overprovisioning Needs Space Reclamation

If your array supports thin provisioning but not space reclamation, you are on a one-way street to a dead end. Yes, you will avoid the up-front capacity wasting of thick volumes. However, your thin volumes will quickly suffer from bloating as data is deleted but not reclaimed by the array. If your array supports thin provisioning and you plan to overprovision, make sure your array also supports space reclamation!

A Space Reclamation Issue to Beware Of

Watch out for potential performance impact in two areas:

- Applications with heavily sequential read patterns
- Performance overhead each time a new page or extent is allocated

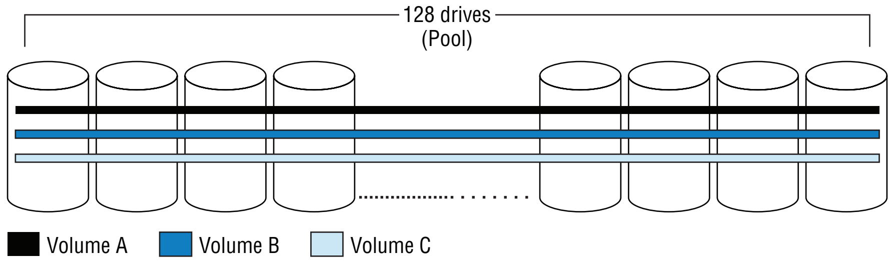
If you are overprovisioning, you need to have eagle eyes on capacity usage and trending of your array. If you take your eye off the ball, you could run out of space and be in a world of hurt.

Go into overprovisioning with your eyes wide open. Overprovisioning does *not* simplify capacity management or storage administration. Anyone who says otherwise probably does not have a lot of hands-on experience with it in the real world.

Pooling and Wide-Striping

Pooling and *wide-striping* are often used synonymously. Both technologies refer to pooling large numbers of drives together and laying out LUNs and volumes over all the drives in the pool. Figure 3.16 shows three volumes wide-striped over all 128 drives in a pool.

FIGURE 3.16 Wide-striped volume layout



It should be said that arrays that do not implement wide-striping should for the most part be considered dinosaurs. All good arrays support pooling and wide-striping.

Pooling and Performance

Pooling has the ability to improve the performance of volumes on your storage array. This is because all volumes potentially have access to all the IOPS and MB/sec of all of the drives on the backend.

Even if pooling doesn't improve the performance of specific volumes, it will reduce hot and cold spots on the backend, ensuring you get the most performance out of the backend of your array. With pooling, you should not see some drives sitting idle while others are being maxed to their limits.

Pooling and Simplified Management

Aside from maximizing and balancing performance, the dawn of wide-striping has greatly simplified storage design and administration. Back in the day, before we had pooling and wide-striping, storage admins and architects would spend hours making sure that two busy volumes didn't use the same four or eight disks on the array, as this would create hot spots and performance bottlenecks. Excel spreadsheets were pushed to their limits to meticulously track and map how volumes were laid out on the backend. Thankfully, this is now a thing of the past!

These days, you land an array on the ground, create a couple of pools, and start creating volumes. You will still probably want to keep data and log volumes on separate pools, but aside from that, you just create your volumes in a pool and let the array balance them. Hot spots on the backend and spindle-based bottlenecks are now extremely rare.

WARNING

Make sure your array supports rebalancing when additional drives are added to the backend. The last thing you want is a nicely balanced backend, where all drives are 80 percent used from a capacity and performance perspective, only to install 128 new drives and find that they are 0 percent used in both respects. This would create a horrendous imbalance in the backend as well as a lot of manual work to rebalance things. All good arrays support rebalancing when new drives are admitted to the configuration.

The Future Is in Pools

Pools and wide-striping also tend to be where all the vendors are putting their innovation. For example, things like thin provisioning, deduplication, and space reclamation are all being built on top of pools and do not work well with traditional RAID groups.

Pools and Noisy Neighbors

Some business-critical applications in some organizations demand isolation, and by design, pooling is the exact opposite of isolation.

Tier 1 line-of-business application owners like to feel safe and that they are the center of the world—and in many cases, as far as the business is concerned, these applications are the center of the world. So lumping their beloved applications on the same set of spindles as every other unworthy tier 3 and tier 4 app just isn't going to give them the warm fuzzies. In these cases, many organizations opt to create ring-fenced pools with dedicated drives and other resources, or even deploy these apps on dedicated arrays and servers.

While pools and wide-striping might not be appropriate for every application, they do work for the vast majority. Speaking from several years' experience in various industry sectors, a rough figure would be that pools are appropriate at least 80 percent to 90 percent of the time. You should rarely, if ever, have to deploy volumes against traditional RAID groups these days. But whatever you do, understand your requirements!

Compression

Compression in the storage industry refers to the process of reducing the size of a data set by finding repeating patterns that can be made smaller. Various complex algorithms can be used to compress data, but the nuts and bolts tend to focus on removing sets of repeating characters and other white space.

Compression of primary storage has never really taken off. Yes, we see it here and there, but the major players have been slow to implement it.

NOTE

Primary storage is storage that is used for data in active use. Even data that is only infrequently accessed and residing on lower tiers is still considered primary storage. Examples of nonprimary storage include archive and backup storage.

Compression of primary storage has not taken off primarily because of its impact on performance. Nobody wants to wait for their primary storage to decompress data. People moan if recalling a three-year-old email from archive takes more than a few seconds, never mind waiting for a long-running query to a volume that has an additional wait included while the storage array decompresses the data being queried.

That said, compression has its place in the storage world, especially in backup and archive use cases and, somewhat interestingly, in SSD-based storage arrays. There are two common approaches to compression:

- Inline
- Post-process

Let's take a look at both before we move on to consider compression's performance impact and the future of compression.

Inline Compression

Inline compression compresses data in cache memory before it hits the drives on the backend. This can obviously use up lots of cache, and if the array is experiencing high volumes of I/O, the process of compressing data can easily lead to slower response times.

On the positive side, though, inline compression can massively reduce the amount of write I/O to the backend. This has the obvious added benefit that less backend disk space is used, but it also means that less internal bandwidth is consumed. The latter is not always considered, but it can be significant.

Post-process Compression

Post-process compression will first land the uncompressed data on the drives on the backend and then at a later time run a background task to compress the data.

Post-processing data takes away the potential performance impact but does nothing to assist with increasing the available internal bandwidth of the array—which, incidentally, can be a major bonus of compression technology. After all, internal bandwidth is often more scarce and always more expensive than disk capacity.

Whether compression is done inline or post process, decompression will obviously have to be done inline and on demand.

Performance Impact

A major performance problem resulting from data compression can be the time is takes to decompress data. Any time a host issues a read request to an array and that data has to be fetched from the backend, this is a slow operation. If you then add on top of that the requirements to decompress the data, you can be looking at a significantly longer operation. This is never ideal. To mitigate this, many storage arrays use the Lempel-Ziv-Oberhumer (LZO) compression algorithm, which doesn't yield the greatest compression in the world, but is simple and fast.

Compression's Future in Primary Storage

With modern CPUs having cycles to burn, and often compression offloads, compression of primary storage is becoming more feasible. Also the use of flash as a storage medium is helping, as the latency incurred when decompressing data is lower because of the faster read speeds of flash, and the usable capacity gains can make flash a more cost-viable option.

Some storage systems are implementing a combination of inline and post-process techniques, performing inline compression when the array can afford to do so, but when I/O levels increase and the going gets tough, they fall back to post-processing the data.

In NAS devices, you should make sure that compression is done at the block level and not the file level. The last thing you want is to have to decompress an entire file if you need to read only a small portion. In fact, in some instances, such as NAS arrays with spinning disk, you may get increased performance if using a good compression technology.

Two final pieces of advice would be, as always, to try before you buy and test before you deploy. In some use cases, compression will work, and in other use cases it will not. Be careful.

Deduplication

Deduplication differs from compression in that deduplication algorithms look for patterns in a data stream that they have seen before—maybe a long time ago. For example, in the manuscript for this book, the word *storage* has been used hundreds of times. A simple deduplication algorithm could take a single instance of the word *storage* and remove every other instance, replacing all other instances with a pointer back to the original instance. If the word *storage* takes up 4 KB and the pointer takes up only 1 KB, we can see where the savings come from. Obviously, that was an overly simplified example.

Whereas compression eliminates small, repeated patterns that are close together (such as the same 32–128 KB chunk), deduplication eliminates larger duplicates over a wider field of data. Deduplication has had a similar history to compression in that it has been widely adopted for archival and backup purposes but has not seen stellar rates of adoption in primary storage. Again, this is mainly due to the potential performance hit.

Like compression, deduplication should always be done at the block level and never at the file level. In fact, file-level deduplication is probably more accurately termed *single instancing*. Obviously, block arrays will do only block-based dedupe, but your NAS vendor could attempt to pull the wool over your eyes by selling you file-level single instancing as *deduplication*. Don't let them do that to you! Deduplication should always be done at the block level, and preferably with a small block size.

Similar to compression, it is generally an inline or a post-process job.

Inline Deduplication

In *inline deduplication*, data is deduplicated on the front end before being committed to disk on the backend.

Inline dedupe requires hashes and then lookups. Strong hashes consume CPU but give good hit/miss accuracy, whereas weaker hashes consume less CPU but require more bit-for-bit lookups (which is slow on spinning disk but feasible on SSD).

Inline deduplication does not require a bloated landing area but may impact front-end performance.

Post-process Deduplication

Data is written to disk in its bloated form and then later is checked to see whether it already exists elsewhere. If it is determined that the data pattern already exists elsewhere on the array, this instance of the data pattern can be discarded and replaced with a pointer.

A drawback to the post-process method is that you need enough capacity on the backend to land the data in its un-deduplicated form. This space is needed only temporarily, though, until the data is deduplicated. This leads to oversized backends.

The benefit of post-process is that it does not impact the performance on the front end.

What to Watch Out For

On NAS arrays, you need to watch out for file-based single instancing being deceivingly branded as deduplication. As noted earlier, the two are not the same. With file-based single instancing, you need two files to be exact matches before you can dedupe. Two 1 MB Word documents that are identical except for a single character in the entire file will not be able to be deduped. Compare that to block level, which will dedupe the entire file except for the few kilobytes that contain the single-character difference. You always want block dedupe, and the smaller the block size, the better!

The scope of deduplication is also important. Ideally, you want an array that deduplicates *globally*. This means that when checking for duplicate data patterns, the array will check for duplicate patterns across the entire array. Checking the entire array gives better probability of finding duplicates than confining your search to just a single volume.

Auto-tiering

Sub-LUN auto-tiering is now a mainstay of all good storage arrays. The theory behind auto-tiering is to place data on the appropriate tier of storage based on its usage profile. This usually translates to data that is being accessed a lot—hot data—residing on fast media such as flash, and data that is rarely accessed—cold data—residing on slower media such as large 7.2K Near Line Serial Attached SCSI (NL-SAS) drives.

Sub-LUN

When it comes to auto-tiering, sub-LUN is where the action is. Early implementations of auto-tiering worked with entire volumes. If a volume was accessed a lot, the entire volume was promoted up to higher tiers of storage. If a volume wasn't accessed very often, the entire volume was demoted to the lower tiers of storage. The problem was that parts of a volume could be very hot, whereas the rest could be cold. This could result in situations where a large volume, say 500 GB, was moved into the flash tier when only 10 MB of the volume was being frequently accessed and the remaining 499.99 GB was cold. That is not great use of your expensive flash tier.

Enter sub-LUN technology. Sub-LUN technology divides a volume into smaller extents, and monitors and moves in extents rather than entire volumes. Assuming our 500 GB example volume from earlier, if the extent size was small enough, only the hot 10 MB of the volume would move up to the flash tier, and the remaining cold data could even filter down to lower tiers. That is a much better solution!

The objective is that if, for example, only 20 percent of your data is active with the remaining 80 percent being relatively inactive, you can fit that 20 percent on high-performance drives, and dump the inactive data sets on cheaper, slower drives.

Sub-LUN Extent Size

As always seems the case, when it comes to extent sizes—whether snapshot extents, thin-provisioning extents, or now sub-LUN extents—smaller is better.

The sub-LUN extent size is the granularity to which a volume is broken up for the purposes of monitoring activity and migrating between tiers. A sub-LUN extent size of 7.6 MB means that the smallest amount of data within a volume that can be moved around the tiers is 7.6 MB.

Generally speaking, high-end enterprise-class arrays have smaller extent sizes than midrange arrays, but this is not always true. However, you should find out the extent size used by your array before purchasing it. This information can usually be easily found in the product documentation or by quickly searching the Internet. Some midrange arrays have a sub-LUN extent size of 1 GB, which can be massively wasteful of premium flash-based resources; nobody in their right mind wants to move 1 GB of data into the flash tier if only 50 MB of that extent is actually busy.

Sizing Up Your Tiers

Most arrays and most tiered storage designs are based around three tiers. These tiers are usually as follows:

Tier 1: Flash/SSD

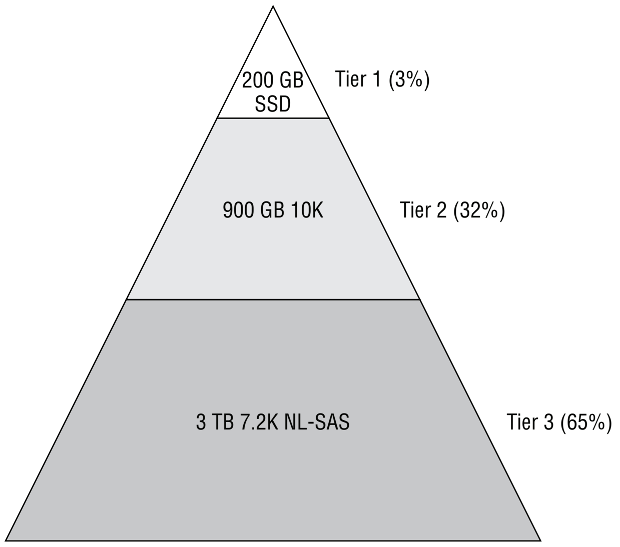
Tier 2: SAS 10K or 15K

Tier 3: NL-SAS 7.2K or 5.4K

How much of each tier to purchase can be highly dependent on your workload, and your vendor or channel partner should have tools available to help you decide.

Generally, the ratios of the three tiers can be represented in a pyramid diagram, as shown in Figure 3.17.

FIGURE 3.17 Pyramid diagram of tier quantities



In Figure 3.17, the percentage values indicate the percentage of capacity and not the percentage of drives. While not attempting to suggest the percentages in the diagram should be used in your environment, they are tailored for a mixed workload array in which the workload parameters were not well known at the time of the design, so it may be a good rough starting point for tuning your own configuration.

Monitoring Period

In order to make decisions about which extents to move up and down the available tiers, the array needs to monitor I/O activity on the backend. It can be important that you monitor the backend performance at the right times. For example, if your organization runs its backups overnight, you *may* want to exclude the backup window from the monitoring schedule so that the backup workloads do not skew the stats and influence the decision-making process. 99 times out of 100, you will not want your backup volumes consuming your higher tiers of storage.

A common approach is to monitor during core business hours on business days, but this may not be appropriate for your environment.

Scheduling Tiering Operations

In addition to deciding when you want to monitor your system, you need to determine data movement windows during which the array is allowed to move extents between the tiers. This, again, will be specific to your business requirements and could be restricted based on how often your array technology allows you to move data. But the point to keep in mind is that moving data about on the backend uses backend resources and, in many cases, cache.

Most companies decide to move their data outside business hours. Often this is during the early hours of the morning, such as between 1 a.m. and 4 a.m., though on some systems continuous motion is a better idea.

Exclusions and Policies

Good storage arrays provide you with a means of including and excluding volumes from auto-tiering operations. For example, you might have some volumes that you have determined need to always reside 100 percent in the middle tier.

You may also have volumes that you want to be controlled by the auto-tiering algorithm but you want to restrict the amount of tier 1 space they consume, or you may want to ensure that they are never relegated to the lowest tier.

Your array should allow you to create tiering polices to achieve fine control of your array's auto-tiering behavior if required.

Auto-tiering with Small Arrays

Auto-tiering solutions tend to be better suited for larger configurations and less so for smaller arrays. One of the main reasons is that each tier needs sufficient drives within it to provide enough performance. If the small array also has a large sub-LUN extent size, this will add to the inefficiencies. Also, licensing costs tend not to make auto-tiering solutions as commercially viable in smaller configurations. From a cost, balancing, and performance perspective, the case for using auto-tiering solutions tends to stack up better in larger arrays.

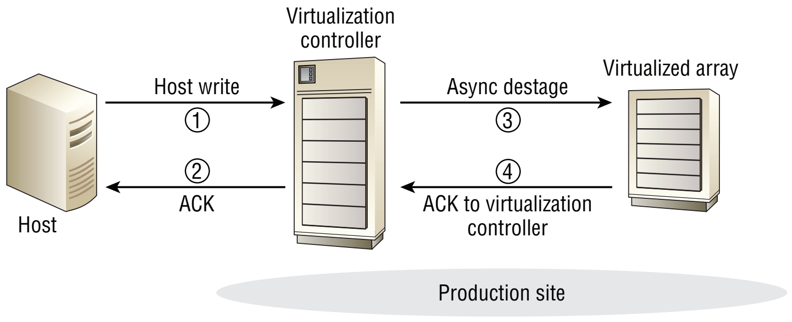
Auto-tiering and Remote Replication

It is easy with some auto-tiering solutions to end up in a situation where your volumes are optimally spread out over the tiers in your source array but not in your target array. This can happen for various reasons, but a major contributing factor is that replication technologies replicate write data only from the source to the target array. Read requests to the source array heavily influence the layout of a volume on the source array. However, the effects of these reads are not reflected on the target array, and hence the replica volume on the target array could well occupy a larger percentage of the lower tiers on the target array. Obviously, if you need to invoke DR, you could see a performance drop due to this.

Storage Virtualization

In this chapter, *storage virtualization* refers to *controller-based virtualization*. Controller-based virtualization is the virtualization of one storage array behind another, as shown in Figure 3.18. In Figure 3.19 later in this section, you can see that write I/Os can be ACKed when they reach the cache in the virtualization controller. Once the data is in the cache of the virtualization controller, it can then be destaged to the virtualized array in a lazy fashion.

FIGURE 3.18 Controller-based virtualization



In a storage virtualization setup, one array acts as the master and the other as the slave. We will refer to the master as the *virtualization controller* and the slave as the *virtualized array*.

The virtualization controller is where all the intelligence and functionality resides. The virtualized array just provides RAID-protected capacity. None of the other features and functions that are natively available in the virtualized array are used.

A Typical Storage Virtualization Configuration

Configuring a typical storage virtualization requires several steps:

1. Configuring the array being virtualized
2. Configuring the virtualization controller
3. Connecting the virtualization controller and the virtualized array

Let's look at each of these steps more closely, and then you'll see an example of how they all fit together.

Configuring the Array Being Virtualized

Usually the first thing to do is to configure the array being virtualized. On this array, you create normal RAID-protected volumes and present them as SCSI LUNs to the virtualization controller. LUN masking should be used so that only the WWPNs of the virtualization controller have access to the LUNs. These LUNs don't need any special configuration parameters and use normal cache settings. It is important that you configure any host mode options for these LUNs and front-end ports according to the requirements of the virtualization controller. For example, if the virtualization controller imitates a Windows host, you will need to make sure that the LUNs and ports connecting to the virtualization controller are configured accordingly.

WARNING

It is vital that no other hosts access the LUN that are presented to the virtualization controller. If this happens, the data on those LUNs will be corrupted. The most common storage virtualization configurations have all the storage in the virtualized array presented to the virtualization controller. This way, there is no need for any hosts to connect to the virtualized array.

Configuring the Virtualization Controller

After configuring the array, you configure the virtualization controller. Here you configure two or more front-end ports—yes, front-end ports—into *virtualization mode*. This is a form of *initiator* mode that allows the ports to connect to the front-end ports of the array being virtualized and discover and use its LUNs. To keep things simple, the ports on the virtualization controller that connect to the array being virtualized will emulate a standard Windows or Linux host so that the array being virtualized doesn't need to have any special host modes configured.

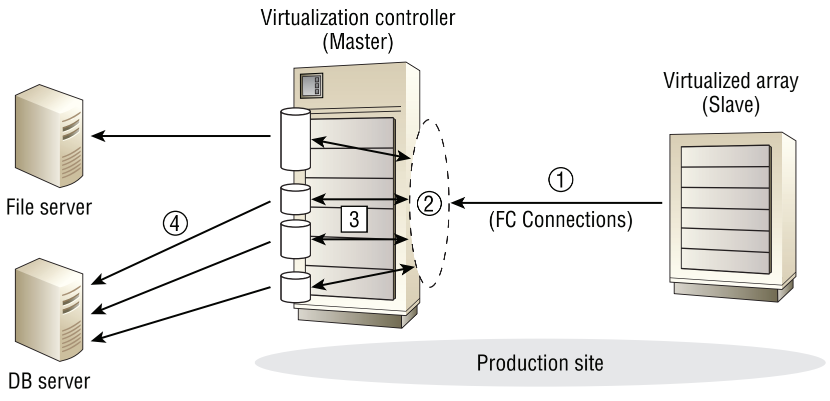
Connecting the Virtualization Controller and the Virtualized Array

Connectivity is usually FC and can be direct attached or SAN attached. Because of the critical nature of these connections, some people opt for direct attached in order to keep the path between the two arrays as simple and clean as possible.

Once the virtualization controller has discovered and claimed the LUNs presented from the virtualized array, it can pretty much use the discovered capacity the same way it would use capacity from locally installed drives. One common exception is that the virtualization controller usually does not apply RAID to the discovered LUNs. Low-level functions such as RAID are still performed by the virtualized array.

Figure 3.19 shows a virtualized array presenting five LUNs to the virtualization controller. The virtualization controller logs in to the virtualized array, discovers and claims the LUNs, and forms them into a pool. This pool is then used to provide capacity for four newly created volumes that are presented out of the front end of the virtualization controller to two hosts as SCSI LUNs.

FIGURE 3.19 Storage virtualization steps



1. LUNs on the virtualized array are presented out to the WWPNs of the virtualization controller.
2. The virtualization controller claims these LUNs and uses them as storage. In this example, they are formed into a pool.
3. Volumes are created from the Pool created in step 2.
4. The volumes created in step 3 are presented out of the front-end ports to hosts as LUNs.

The local storage in the virtualization controller is referred to as *internal storage*, whereas the storage in the array being virtualized is referred to as *external storage* or *virtualized storage*.

Putting It All Together

To understand all the stages of configuring storage virtualization, it is probably best to look at an example of the steps in order. Exercise 3.4 explores this.

EXERCISE 3.4

Configuring Storage Virtualization

The following steps outline a fairly standard process for configuring storage virtualization. The exact procedure in your environment may vary, so be sure to consult your array documentation, and if in doubt consult with your array vendor or channel partner.

On the array being virtualized, perform the following tasks:

1. Carve the backend storage into RAID-protected volumes. Usually, large volumes, such as 2 TB, are created.
2. Present these LUNs on the front-end ports and LUN-mask them to the WWPNs of the virtualization controller.
3. Configure any standard cache settings for these LUNs.
4. Configure any host mode settings required by the virtualization controller. For example, if your virtualization controller emulates a Windows host, make sure you configure the host settings appropriately.

On the virtualization controller, perform the following tasks:

5. Configure a number of front-end ports as *external ports* (Virtualization mode). You should configure a minimum of two for redundancy.

Now that the basic tasks have been performed on the virtualization controller and the array has been virtualized, the arrays need to be connected. These connections can be either direct-attached or SAN-attached connections. Once the arrays are connected and can see each other, perform the following steps on the virtualization controller:

6. Using the virtualization controller's GUI or Command Line Interface (CLI), discover the LUNs presented on the virtualized array.
7. Configure the newly discovered LUNs into a pool.

It can take a while for the LUNs to be discovered and imported on the virtualization controller, and it can take a while for the pool to be created. Once the pool is created, it can usually be used as if it was a nonvirtualized pool using internal disks. From this point, volumes can be created and bound to the new pool. These volumes can be either mapped to hosts on the front end or used to create a pool.

Prolonging the Life of an External Array

One of the potential reasons for virtualizing an array is to prolong its life. Instead of turning the old array off because it's out-of-date and doesn't support the latest advanced features we require, we can plug it into our shiny new array and give it a brain transplant.

This idea works in principle, but the reality tends to be somewhat more complicated. For this reason, even in a difficult economy with diminishing IT budgets, arrays are rarely virtualized to prolong their life.

Some of the complications include the following:

- Having to keep the virtualized array on contractual maintenance congruent to the environment it will continue to serve. It is not the best idea in the world to have the old virtualized array on Next Business Day (NBD) response if it's still serving your tier 1 mission-critical apps.
- Multivendor support can be complicated if you are virtualizing an array from a third-party vendor. This is made all the more challenging when the array you are virtualizing is old, as the vendor is usually not keen for you to keep old kit on long-term maintenance, and the vendor generally stops releasing firmware updates.

Experience has shown that although virtualizing an array to prolong its life works on paper, the reality tends to be somewhat messy.

Adding Functionality to the Virtualized Array

It's common for customers to buy a new tier 1 enterprise-class array along with a new tier 2/3 midrange array and virtualize the midrange array behind the enterprise-class array. This kind of configuration allows the advanced features and intelligence of the tier 1 array to be extended to the capacity provided by the virtualized tier 2 array. Because the array performing the virtualization treats the capacity of the virtualized array the same way it treats internal disk, volumes on the virtualized array can be thin provisioned, deduplicated, replicated, snapshotted, tiered, hypervisor offloaded…you name it.

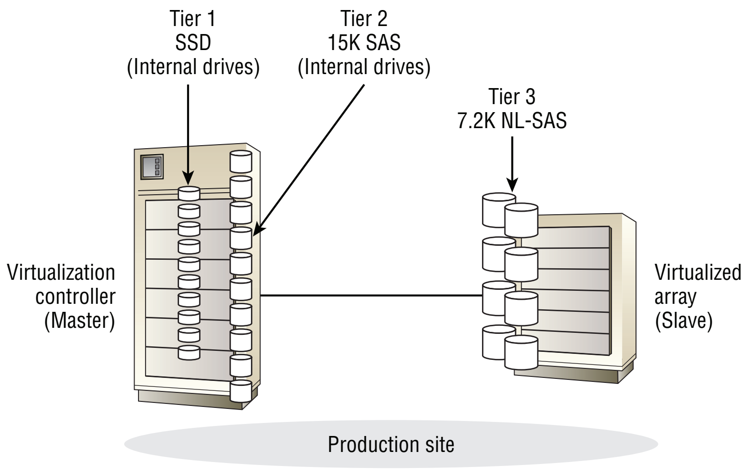
Storage Virtualization and Auto-tiering

All good storage arrays that support controller-based virtualization will allow the capacity in a virtualized array to be a tier of storage that can be used by its auto-tiering algorithms. If your array supports this, then using the capacity of your virtualized array as your lowest tier of storage can make sound financial and technical sense.

Because the cost of putting disk in a tier 1 array can make your eyes water—even if it's low-tier disk such as 4 TB NL-SAS—there is solid technical merit in using only the internal drive slots of a tier 1 array for high-performance drives. Aside from the cost of loading internal drive slots, they also provide lower latency than connecting to capacity in a virtualized array. So it stacks up both financially and technically to keep the internal slots for high-performance drives.

This leads nicely to a multitier configuration that feels natural, as shown in Figure 3.20.

FIGURE 3.20 Storage virtualization and auto-tiering



There is no issue having the hot extents of a volume on *internal* flash storage, the warm extents on *internal* high performance SAS, but having the coldest and least frequently accessed extents down on *external* 7.2 K or 5.4 K NL-SAS. This kind of configuration is widely deployed in the real world.

Of course, this all assumes that your virtualization license doesn't cost an arm and a leg and therefore destroy your business case.

Virtualization Gotchas

There is no getting away from it: storage virtualization adds a layer of complexity to your configuration. This is not something that you should run away from, but definitely something you should be aware of before diving in head first.

Real World Scenario

Complications That Can Come with Storage Virtualization

A company was happily using storage virtualization until a well-seasoned storage administrator made a simple mistake that caused a whole swath of systems to go offline. The storage administrator was deleting old, unused *external* volumes—volumes that were mapped from the virtualized array through the virtualization controller. However, the virtualization controller gave volumes hexadecimal numbers, whereas the virtualized array gave volumes decimal numbers. To cut a long story short, the administrator got his hex and decimal numbers mixed up and deleted the wrong volumes. This resulted in a lot of systems losing their volumes and a long night of hard work for several people. This complexity of hex and decimal numbering schemes would not have existed in the environment if storage virtualization was not in use.

Depending on how your array implements storage virtualization, and also sometimes depending on how you configure it, you may be taking your estate down a one-way street that is hard to back out of. Beware of locking yourself into a design that is complicated to unpick if you decide at a later date that storage virtualization is not for you.

If your array is not designed and sized properly, performance can be a problem. For example, if you don't have enough cache in your virtualization controller to deal with the capacity on the backend, including capacity in any virtualized arrays, you may be setting yourself up for a bunch of performance problems followed by a costly upgrade. Also, if the array you are virtualizing doesn't have enough performance—usually not enough drives—to be able to destage from cache quickly enough, you can make the cache of your virtualization head a choking point.

While on the topic of performance, by far the most common implementation of storage virtualization these days is using the virtualized array for cheap and deep tier 3 storage. It is rare to see customers loading virtualized arrays up with high-performance drives.

Finally, cost can be an issue if you don't know what you're getting yourself into. Virtualization licenses rarely come for free.

Hardware Offloads

Hardware offloads, sometimes referred to as *hardware accelerations,* are a relatively recent trend in the open systems storage world. A *hardware offload* is the act of executing a function in dedicated hardware rather than software. In respect to storage, this usually translates into operating systems and hypervisors offloading storage-related functions such as large copy jobs and zeroing out to the storage array.

There are a few objectives in mind when implementing hardware offloads to a storage array. These tend to be as follows:

- Improve performance
- Reduce the load on the host CPU
- Reduce the load on the network

While it's true that most servers these days have CPU cycles to burn, many storage-related tasks can still be performed faster and more efficiently by offloading them to the storage array.

Let's look at a few of the more popular examples in the real world.

VMware VAAI

VMware was perhaps the first vendor in the open systems world to push the notion of hardware offloads. VMware's vStorage API for Array Integration (VAAI) is a suite of technologies designed to offload storage-related operations to a VAAI-aware storage array. There are VAAI offloads for both block storage and NAS storage, referred to as *block primitives* and *NAS primitives*, respectively.

Block storage primitives are based on T10 SCSI standards and are natively supported by any storage array that supports those T10 standards, although be sure to check the

VMware Hardware Compatibility List (HCL) before making any assumptions. VAAI for NAS, on the other hand, requires a plug-in from your NAS vendor.

NOTE Despite the name—vStorage API for Array Integration—VAAI primitives are not really APIs. Instead the vSphere hypervisor is simply written to understand SCSI standards and SCSI commands that allow it to offload storage-related functions to a storage array that supports the same SCSI commands.

Let's look at each of the available VAAI offloads.

ATS

If you know anything about Virtual Machine File System (VMFS), you will know that certain metadata updates require a lock to ensure the integrity of the updated metadata and the entire VMFS volume.

Back in the day, the only locking mechanism that was available was a SCSI reservation. And the problem with a SCSI reservation is that it locks an entire LUN, meaning that any time you made a metadata update you had to lock the entire LUN. And when a LUN is locked, it cannot be updated by any host other than the host that issued the reserve command. That is not the end of the world if we're talking about small, lightly used VMFS volumes. However, this could become a problem with large, heavily used VMFS volumes accessed by multiple hosts. This is where atomic test and set (ATS) comes to the rescue!

ATS, also known as *hardware-assisted locking,* brings two things to the party:

Extent-Based Locking Extent-based locking means that you no longer have to lock up the entire LUN with a SCSI reserve when updating the metadata in a VMFS volume. You need to lock only the extents that contain the metadata being updated.

More-Efficient Locking Procedure The act of engaging and releasing the extent-based lock is offloaded to the storage array, and the locking mechanism requires fewer steps, making it quicker and more efficient.

Both of these added significantly to the scalability of VMFS and were major factors behind the ability to deploy larger VMFS datastores.

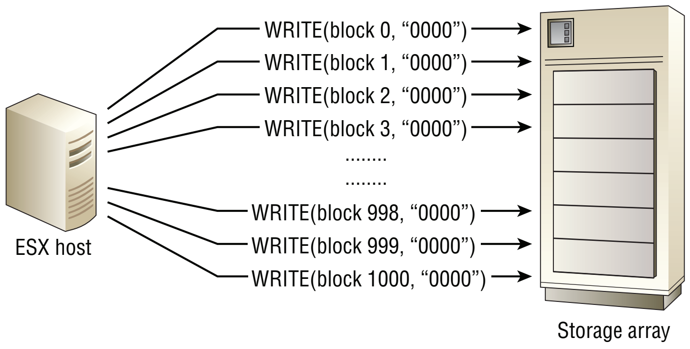
Since the initial release of VAAI, ATS has now been formalized as a T10 SCSI standard using SCSI opcode 0x89 and the `COMPARE_AND_WRITE` command. This ensures standards-based implementations by all supporting storage arrays.

Hardware-Accelerated Zeroing

Certain situations in VMware environments require the zeroing out of entire volumes. For example, eager zeroed thick (EZT) volumes are volumes that are wiped clean by writing zeros to every sector in the volume. This has positive security implications by ensuring that no data from a volume's previous life can inadvertently be visible. It also has potentially positive performance implications by not requiring ESX to have to zero a block the first time it writes to it—although don't expect the performance improvements to blow you away.

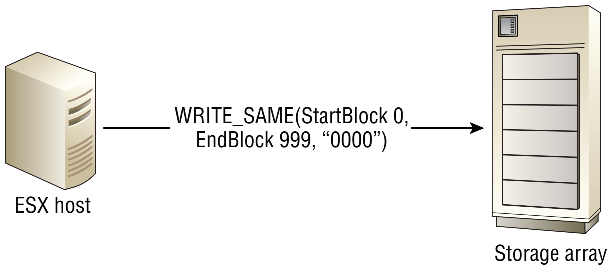
Let's look at a quick example of hardware-accelerated zeroing in action. Assume a volume with 1,000 blocks and that you need to zero out this entire volume. Without hardware-accelerated zeroing, ESX would have to issue the set of commands outlined in Figure 3.21.

FIGURE 3.21 Zeroing out a volume without VAAI



With the introduction of hardware-accelerated zeroing, the exact same outcome can be achieved via a single command, as shown in Figure 3.22.

FIGURE 3.22 Zeroing out a volume with VAAI



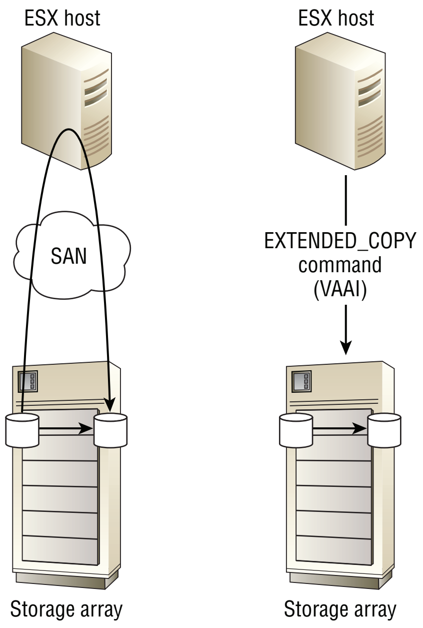
Not only does this offload the operation from the ESX host, freeing up CPU cycles, it also vastly reduces network chatter, as the ESX host can issue a single command and the array can give a single response. In addition, the storage is probably able to do the work faster. The net result is that the operation completes faster and uses less ESX CPU and less network resources.

Hardware-accelerated zeroing works via the T10 SCSI standard `WRITE_SAME` command (opcode 0x93).

Hardware-Accelerated Copy

Popular VMware-related operations, such as Storage vMotion and creating new VMs from a template, utilize large data-copy operations. Hardware-accelerated copy uses the SCSI EXTENDED_COPY command (opcode 0x83) to offload the heavy lifting work of the data copy to the storage array. Figure 3.23 shows a large copy operation without and with the use of EXTENDED_COPY.

FIGURE 3.23 Large copy command without and with hardware-assisted offload



In Figure 3.23, the operation on the left that does not use the EXTENDED_COPY offload requires all data being copied from the source volume to the target volume to pass up from the array, through the SAN, through the HBAs in the ESX host, through the kernel data mover engine, back out through the HBA, back down through the SAN, and back to the storage array. In contrast, the operation on the right shows a single EXTENDED_COPY command issued from the ESX host to the array. The array then copies the data internally without it having to pass all the way up the stack, through the host, and back down again.

With the use of the EXTENDED_COPY hardware assist, network bandwidth utilization can be dramatically decreased, as can host-based resource utilization.

Thin Provisioning Stun

Thin provisioning stun (TP Stun) is an attempt to gracefully deal with out-of-space conditions on overprovisioned storage arrays. TP stun is a mechanism by which virtual machines that request additional space from a storage array that has run out space can be paused rather than crash. This is achievable because the storage array can now inform the host that its disk is full rather than just issuing a write fail to the host, allowing the host to more gracefully deal with the scenario. Once the array has more space, the condition is cleared and VM can be resumed.

While this might not sound perfect, it's an attempt to make the most of what is a nightmare scenario. This is of course relevant only on thin-provisioning storage arrays that are overprovisioned and VMs that reside on datastores that are thinly provisioned from the storage array.

VAAI and Zero Space Reclamation

The latest versions of VMware are now fully aware of array-based thin provisioning and optimized assist in the ongoing effort to keep thin volumes thin.

VAAI now utilizes the T10 SCSI standard `UNMAP` command to inform the storage array that certain blocks are no longer in use. Examples include deleted VMs or VMs that have been migrated to other datastores.

Full File Clone for NAS

Full file clone for NAS brings offloading the cloning of virtual disks to the VMware NAS party. This feature is similar to the functionality of `EXTENDED_COPY` in the block world, in that it offloads VM copy operations to the NAS array. However, it is not quite as functional as its block counterpart, as it can be used for only clone operations and cannot be used for Storage vMotion (meaning that the VM must be powered off). Storage vMotion operations still revert to using the VMkernel's software data mover.

Fast File Clone for NAS

Fast file clone for NAS brings support for offloading VM snapshot creation and management to the native snapshot capabilities of the NAS array. This enables snapshots to be created much quicker and with less network-related traffic.

No ATS Needed for NAS

Interestingly, there is no need for the equivalent of an ATS primitive in the NAS world, as the NAS world has never been plagued with rudimentary locking systems such as the `SCSI RESERVE` mechanism. This is because NAS arrays understand and own the underlying exported filesystem and therefore have native advanced file-level locking capabilities.

Microsoft ODX

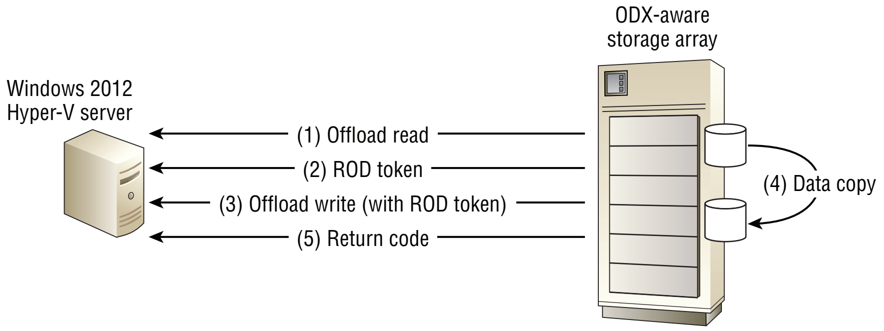
With Windows 8 and Server 2012, Microsoft introduced *Offloaded Data Transfer (ODX)* to the world. ODX is a data-copy offload technology, designed to offload data-copy functions to intelligent storage arrays in much the same way that VMware does with VAAI. It could be said that ODX is to Windows what VAAI is to VMware. The difference is that VMware has been doing it for longer, and therefore has more capabilities and more maturity, though this will no doubt level out over time.

As ODX is similar to some VAAI primitives, it exists to speed up certain storage-based operations such as large copy operations and bulk zeroing, while at the same time saving on host CPU, NIC utilization, and network bandwidth.

ODX for Large File Copies

Figure 3.24 shows how ODX works for a large copy operation.

FIGURE 3.24 High-level ODX copy operation



In the large copy operation represented in Figure3.24, the host running Windows Server 2012, or later, issues an *offload read* command to the ODX-aware array. The array responds with a 512-byte token that is an array-specific logical representation of the date to be copied. This token is known as a *representation of data (ROD)* token. The host then issues an *offload write* request, with the associated ROD token, to the array. The array then performs the copy without the data ever leaving the array.

ODX can also work between two arrays that both support this particular ODX configuration.

ODX works with the following technologies:

- Hyper-V virtual hard disks (VHD)
- SMB shares (sometimes referred to as CIFS)
- Physical disks
- FC, iSCSI, FCoE, SAS

ODX will be invoked to perform the data copy any time a copy command is issued from the CLI, PowerShell prompt, Windows Explorer, or Hyper-V migration job, as long as the server is running Windows Server 2012 or later and the underlying storage array supports ODX. However, things like deduplication and Windows BitLocker-encrypted drives do not work with the initial release of ODX. Future versions of these filter drivers will potentially be ODX aware.

ODX utilizes `T10 XCOPY LITE` primitives, making it standards based.

As with all developing technologies, it is spreading its wings. Check with vendors and Microsoft for supported configurations. Things to beware of include Resilient File System (ReFS), dynamic disks, and similar developments.

ODX and Bulk Zeroing

ODX can also be used to perform bulk zeroing by using the *Well Known Zero Token* ROD token.

NOTE A ROD token can be a vendor-specific 512-byte string that, for example, represents the data range to be copied. Alternatively, a ROD token can be a Well Known token such as the Zero Token that is used to perform bulk zero operations.

Chapter Essentials

Dual Controller and Grid Architectures Dual-controller architectures, sometimes referred to as midrange, provide many of the advanced features seen in enterprise-class grid architectures but at a cheaper price point. Dual-controller architectures are also limited in scalability and do not deal with hardware failures as well as grid architectures. Grid architectures offer scale-out and deal better with hardware failures, but at a higher cost.

Redundancy Redundancy is a hallmark of storage designs. Most storage arrays employ redundancy at every level possible, ensuring that failed components do not interrupt the operation of the array. Even at the host level, multiple paths are usually configured between the host and storage in multipath I/O (MPIO) configurations, ensuring that the loss of a path or network link between the host and storage array does not take the system down.

Replication Array-based replication makes remote copies of production volumes that can play a vital role in disaster recovery (DR) and business continuity (BC) planning. Depending on your application and business requirements, remote replicas can either be zero-loss bang-up-to-date synchronous replicas or they can lag slightly behind, using asynchronous replication technology. Asynchronous replication technologies can have thousands of miles between the source and target volumes, but synchronous replication requires the source and target to be no more than ~100 miles apart.

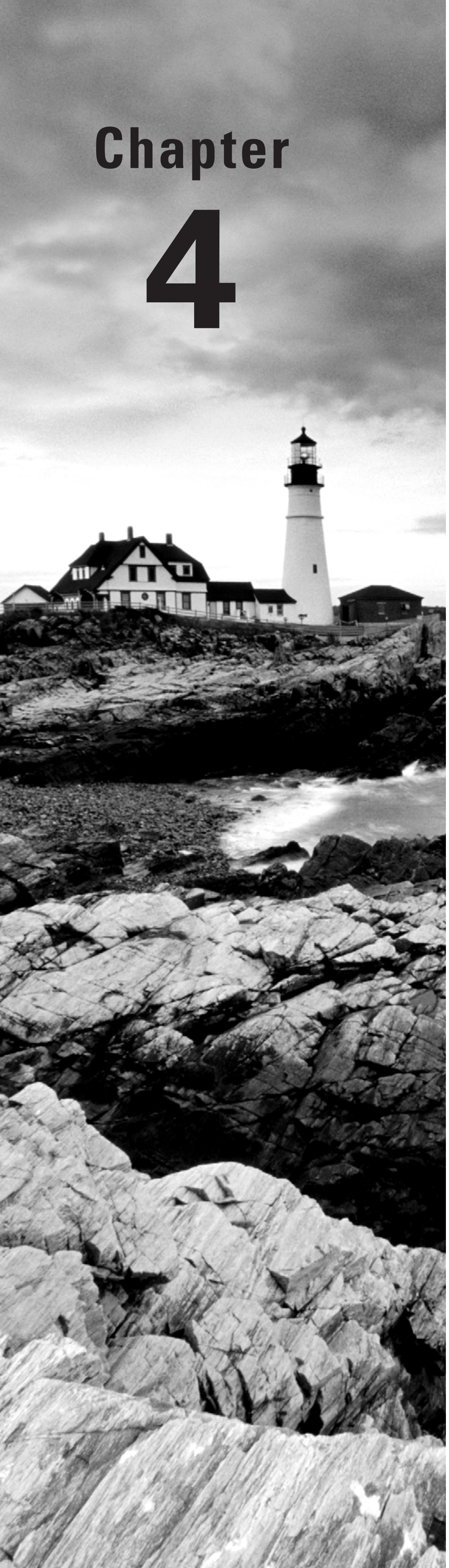
Thin Provisioning Thin-provisioning technologies can be used to more effectively utilize the capacity in your storage arrays. However, if overprovisioning an array, you must take great care to ensure that your array does not run out of available space.

Sub-LUN Tiering Sub-LUN tiering technologies allow you to place data on the most appropriate tier of storage—frequently accessed data on fast media, and inactive data on slow media. This can improve the performance of your array and bring costs down by not having to fill your array with fast disks when most of your data is relatively infrequently accessed.

Summary

In this chapter we covered all things storage array related. We discussed; architectural principles, including the front-end, the backend, the importance of cache, different types of persistent media, and LUNs and volumes. We even touched on snapshots and clones, as well as replication technologies, though these will be covered in more detail in their own dedicated chapters of this book. We talked about pros and cons, use cases, and impact and disruptive nature of flash-based solid state media.

We also talked about some of the advanced features, and so-called intelligence, of storage arrays, including tiering, deduplication, and hardware offloads, many of which integrate with server- and hypervisor-based technologies.



Chapter 4

RAID: Keeping Your Data Safe

TOPICS COVERED IN THIS CHAPTER:

- ✓ **What RAID is and why we need it**
- ✓ **Hardware vs. software RAID**
- ✓ **Striping**
- ✓ **Parity**
- ✓ **Mirroring**
- ✓ **Standard RAID levels**
- ✓ **Performance**
- ✓ **Reliability**
- ✓ **The future of RAID**

This chapter covers all of the RAID material required for the CompTIA Storage+ exam, and more. You'll start with the basics, including a history of RAID technology and why it has been, and continues to be, such a fundamental requirement in enterprise technology across the globe.

This chapter compares different implementations of RAID, including an analysis of both hardware RAID and software RAID, the pros and cons of each, and some potential use cases for both. It also covers the fundamental principles and concepts that comprise most of the RAID implementations commonly seen in the real world, including RAID groups, striping, parity, mirroring, and the commonly implemented RAID levels.

You'll finish the chapter looking at some of the more recent and niched implementations, as well as what the future may hold for this long-standing technology that might be starting to struggle in coping with the demands of the modern world.

The History and Reason for RAID

Back in the day, big expensive computers would have a single, large, expensive drive (SLED) installed. These disk drives

- Were very expensive
- Posed a single point of failure (SPOF)
- Had limited IOPS

RAID was invented to overcome all of these factors.

NOTE In 1988, Garth Gibson, David Patterson, and Randy Katz published a paper titled "A Case for Redundant Arrays of Inexpensive Disks (RAID)." This paper outlined the most commonly used RAID levels still in use today and has proven to be the foundation of protecting data against failed disks for over 20 years. As their work was published as a *paper* and not a *patent*, it has seen almost universal uptake in the industry and remains fundamental to data protection in data centers across the globe.

Fast forward 25+ years. Disk drives still fail regularly and often without warning. Most large data centers experience multiple failed disk drives each day. And while disk drive capacities have exploded in recent years, performance has not, leading to a disk drive bloat problem, where capacity and performance are severely mismatched—although solid-state media is changing the game here a little. These facts combine and conspire to ensure that RAID technology remains fundamental to the smooth running of data centers across the world.

So, even though RAID technology is over 25 years old, a solid understanding of how it works and protects your data will stand you in good stead throughout your IT career.

TIP

RAID is a form of *insurance policy* for your data, and you do not want to find yourself uninsured in a time of crisis. Yes, RAID has a cost, but that cost is literally pennies when compared to the cost of downtime or application rebuilds. Also, your insurance portfolio for data protection should include far more than just RAID! Your business-critical applications should have multiple levels of protection, including at least the following: RAID, MPIO, replication, and backups.

What Is RAID?

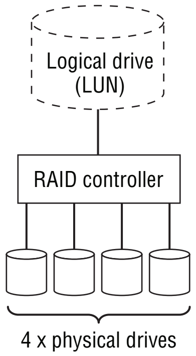
RAID is an acronym. Back in the day it referred to *redundant array of inexpensive disks*. Nowadays, most folks refer to it as redundant array of *independent* disks. And with the emergence of solid-state drives, which aren't really disks, maybe it's about to see another change and become redundant array of independent *drives*.

Anyway, the raison d'etre of RAID technology in the modern IT world is twofold:

- Protect your data from failed drives.
- Improve I/O performance by parallelizing I/O across multiple drives.

In order to deliver increased performance and protection, RAID technology combines multiple physical drives and creates one or more *logical drives* that span those multiple physical drives, as shown in Figure 4.1.

FIGURE 4.1 Logical drive created from four physical drives



A logical drive can typically be one of two things. It can be a subset of a single physical drive with the intention of creating multiple logical drives—sometimes called *partitions*—from the single physical drive. Or, as is the case in Figure 4.1, a logical drive can be a device created from capacity that is aggregated from multiple physical drives. Figure 4.1 shows a single logical drive created from capacity on four physical drives. Sometimes a logical drive can be referred to as a *logical disk*, *logical volume*, *virtual disk*, *or virtual volume*.

There are several *RAID levels* in wide use today—such as RAID 1, RAID 10, RAID 5, and RAID 6—each of which has its own strengths and weaknesses. You'll look closely at the more popular RAID levels throughout the chapter and consider their strengths and weaknesses, as well as potential use cases. However, before looking closer at RAID levels, let's first look at some of the fundamental concepts and components.

WARNING

Actually, before we set out, it is of paramount importance to stress the following: RAID is not a backup solution or a replacement for backups! RAID protects data only against disk drive failures and not the many other things that can destroy your data.

Hardware or Software RAID

RAID can be implemented in software or hardware.

Software RAID is usually implemented as part of the OS, on top of the OS by a *logical volume manager* such as Linux LVM, or as part of a filesystem as with ZFS. Either way, this results in software RAID using host resources such as CPU and RAM. This is not as much of a problem as it once was, thanks to the monster CPUs and oodles of RAM we commonly see in servers these days. Consumption of host-based resources is not much of a problem if implementing RAID 1 mirroring, but it can become more of an issue when performing parity RAID such as RAID 5 or RAID 6, especially if you have a lot of drives in the RAID set. Also, because the OS needs to be booted before the RAID software can initialize, OS boot volumes cannot be RAID protected with software RAID. Throw into the mix the lack of some of the fringe benefits that typically come with hardware RAID, such as faster write performance and faster rebuilds, and software RAID quickly starts to look like a poor-man's RAID.

Microsoft Storage Spaces In the Windows Server 2012 operating system, Microsoft has implemented an LVM known as *Storage Spaces*.

Storage Spaces allows you to pool together heterogeneous drives and create virtual volumes, known as *storage spaces*, using the capacity in these pools. These storage spaces can optionally be software RAID protected—either mirror protected (RAID 1) or parity protected (RAID 5).

While Storage Spaces is a relatively recent attempt at providing advanced storage features in OS software, its implementation of software RAID still suffers from the same performance penalties that other software RAID solutions suffer from when it comes to using parity for protection. Also, Storage Spaces offers none of the advanced filesystem and software RAID integration that ZFS does.

While the pooling aspects of Storage Spaces might be good, the software RAID implementation is basic and has not been able to reach escape velocity, remaining grounded to the same performance and usage limitations found in so many other software RAID implementations.

NOTE Mirroring, parity, and RAID levels are explained in more detail in the "RAID Concepts" section.

Hardware RAID, on the other hand, sees a physical *RAID controller* implemented in the server hardware—on the motherboard or as an expansion card—that has its own dedicated CPU and either battery-backed or flash-backed RAM cache. Because of this dedicated hardware, no RAID overhead is placed on the host CPU. Also, because hardware RAID controllers initialize before the OS, this means that boot volumes can be RAID protected. The battery-backed or flash-backed cache on the RAID controller also offers improved write performance via *write-back caching*—acknowledging the I/O back to the host when it arrives in cache rather than having to wait until it is written to the drives in the RAID set. One potential downside to hardware RAID controllers inside servers is that they provide a SPOF. Lose the RAID controller, and you lose your RAID set. Despite this, if you can afford it, most people find that they go with hardware RAID most of the time.

Table 4.1 highlights the pros and cons of hardware and software RAID.

TABLE 4.1 Hardware vs. software RAID

Feature	Hardware RAID	Software RAID
Host CPU offload	Yes	No
RAID for boot volumes	Yes	No
Write-back cache	Yes	No
Hot-swap drives	Yes	Not always
Complexity	Lower	Higher
Cost	High	Low
Flexibility	Lower	Higher

External RAID controllers, such as storage arrays, are also forms of hardware RAID. Storage arrays usually have several advantages over and above stand-alone hardware RAID controllers that are typically installed in server hardware. These advantages include the following:

- Redundancy. RAID controllers installed on server motherboards or as PCIe cards are single points of failure. While it is true that storage arrays can themselves be single points of failure, they tend to be extremely highly available, and it is rare that they catastrophically fail.
- Storage arrays offer larger caches, so they can often achieve higher performance.
- Storage arrays support multiple drive types. Internal RAID controllers tend not to allow you to mix and match drives.
- Storage arrays support many more drives for both capacity and increasing performance through parallelization.
- Storage arrays usually offer advanced features such as snapshots, replication, thin provisioning, and so forth.

Microsoft Storage Spaces—Software RAID

In the Windows Server 2012 operating system, Microsoft has implemented an LVM known as *Storage Spaces*.

Storage Spaces allows you to pool together heterogeneous drives and create virtual volumes, known as *storage spaces*, using the capacity in these pools. These storage spaces can optionally be software RAID protected—either mirror protected (RAID 1) or parity protected (RAID 5).

While Storage Spaces is a relatively recent attempt at providing advanced storage features in OS software, its implementation of software RAID still suffers from the same performance penalties that other software RAID solutions suffer from when it comes to using parity for protection. Also, Storage Spaces offers none of the advanced filesystem and software RAID integration that ZFS does.

While the pooling aspects of Storage Spaces might be good, the software RAID implementation is basic and has not been able to reach escape velocity, remaining grounded to the same performance and usage limitations found in so many other software RAID implementations.

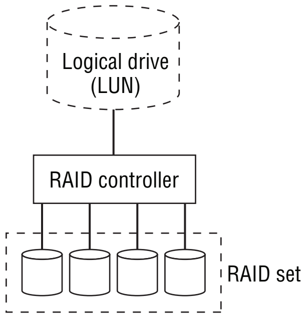
RAID Concepts

Let's tackle the major concepts that comprise most RAID technologies. A solid understanding of these concepts is vital to a full understanding of RAID technologies and the impact they will have on your data availability and performance.

RAID Groups

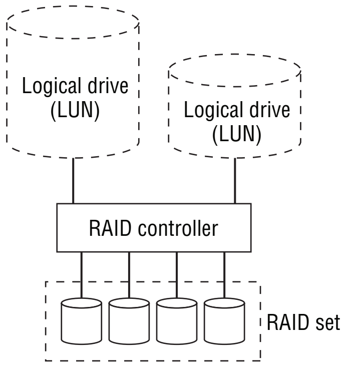
RAID groups, also known as *RAID sets* or *RAID arrays*, is the term used to refer to a group of drives that are combined and configured to work together to provide increased capacity, increased performance, and increased reliability. All drives in a RAID group are connected to the same RAID controller and are owned by that RAID controller. Figure 4.2 shows a RAID set containing four drives that has been configured as a single logical drive.

FIGURE 4.2 RAID set configured as a single logical drive



It is also possible to configure more than one logical volume per RAID set. Figure 4.3 shows the same four-disk RAID set, but this time configured as two logical drives of differing sizes.

FIGURE 4.3 RAID set configured as two logical drives



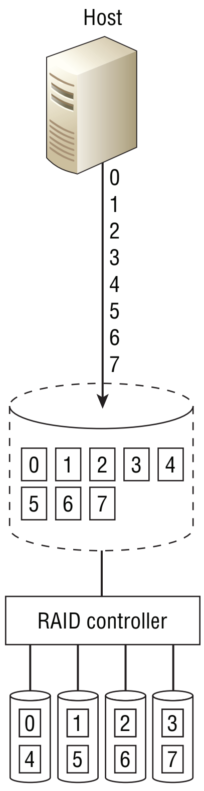
Drives in a RAID set are sometimes referred to as *members*.

Striping

The process of creating a logical drive/logical volume that is spread across all the drives in a RAID set is known as *striping*. The idea behind striping is to parallelize I/O across as many drives as possible in order to get the performance of every drive. Every write to a logical drive that is striped across all drives in a RAID set gets distributed across all the drives in the RAID set, potentially giving it access to all of the IOPS and MB/sec of every drive.

Figure 4.4 shows I/O from a host being written to a logical volume that is striped over four physical drives. If each physical drive is capable of 200 IOPS, the potential performance from the logical volume is 800 IOPs. If the volume was not striped, but instead confined entirely to a single drive, it could perform at only 200 IOPS.

FIGURE 4.4 I/O being striped across four physical drives



Striping is a technique that is fundamental to many RAID algorithms for both performance and reliability reasons.

The process of striping a logical drive over multiple physical drives also allows the logical drive to access all the *capacity* available on each physical drive. For example, a logical volume created on a simple RAID set containing four 900 GB drives could be as large as 3,600 GB. Thus, striping also allows for increased capacity as well as performance.

Parity

Parity is a technique used in RAID algorithms to increase the *resiliency*, sometimes referred to as *fault tolerance*, of a RAID set.

Parity-based RAID algorithms reserve a percentage of the capacity of the RAID set (typically one or two drives' worth) to store parity data that enables the RAID set to recover from certain failure conditions, such as failed drives.

Let's look at a quick and simple example of how parity works to protect and reconstruct binary data. Parity can be either even or odd parity, and it doesn't matter which of the two is employed, as they both achieve the same end goal: providing fault tolerance to binary data sets. Let's use odd parity in our examples. *Odd parity* works by making sure that there is always an odd number of ones (1s) in a binary sequence. Let's assume all of our binary sequences comprise four bits of data, such as 0000, 0001, 1111, and so on. Now let's add an extra bit of data after the fourth bit, and use this extra bit as our parity bit. As we are using odd parity, we will use this bit to ensure that there is always an odd number of 1s in our five bits of data. Let's look at a few examples:

0000**1**—The fifth bit of data (our parity bit) is a 1 so that there is an odd number (one) of 1s in our data set.

0111**0**—This time we make the fifth bit of data a 0, to ensure there are an odd number of 1s (three) in the data set.

1000**0**—This time we make the parity bit a 0 so that there is an odd number of 1s (one) in the data set. If we'd made it a 1, there would have been two 1s in the data set.

All well and good, but how does this make our data set fault tolerant? First up, as we've added only a single parity bit, our data set can tolerate the loss of only a single bit of data, but that lost bit can be any bit in the data set. Anyway, to see how it works, let's look at our three example data sets again, only this time we've removed a bit of data and will use the principle of odd parity to determine and reconstruct that missing bit of data.

0_001—As we are using odd parity, our missing bit must be a 0. Otherwise, we would have an even number of 1s in the data set.

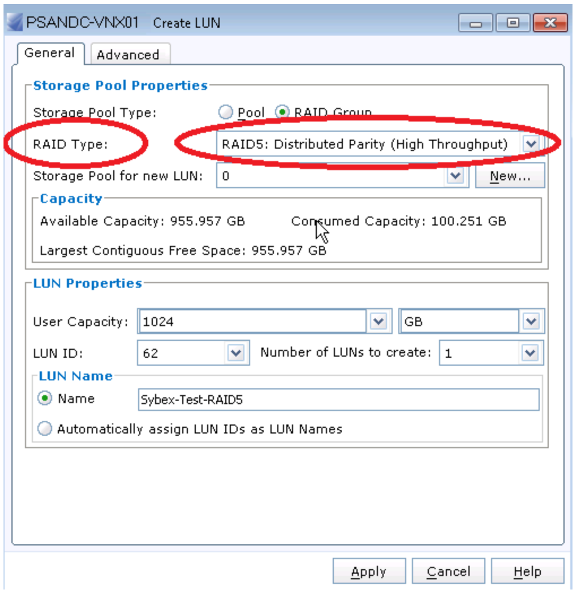
011_0—This time we know that the missing bit is a 1. If it was a 0, we would have an even number of 1s in our data set.

1000_—This time we've lost the parity bit. But this makes no difference. We can still apply the same logic and determine that we are missing a 0.

As our example data set has five bits of data but contains only four bits of real user data—one bit is parity overhead—we have lost one bit, or 20 percent. Put in other terms, our data set is 80 percent efficient, as 80 percent of the bits are used for real data, with 20 percent used to provide fault tolerance. And it may be useful to know that at the low level, parity is computing using an exclusive OR (XOR) operation, although knowledge of this won't help you in your day-to-day job.

Figure 4.5 shows the option to configure a RAID 5 volume on an EMC VNX array. As you can see, the GUI used to create the volume doesn't ask anything about XOR calculations or other RAID internals.

FIGURE 4.5 Creating a RAID 5 volume on an EMC storage array



Storage Pool Properties

Storage Pool Type: Pool [✓ RAID Group]

RAID Type: RAID5: Distributed Parity (High Throughput)

Storage Pool for new LUN: 0

Capacity

Available Capacity: 955.957 GB Consumed Capacity: 100.251 GB

Largest Contiguous Free Space: 955.957 GB

LUN Properties

User Capacity: 1024 GB

LUN ID: 62 Number of LUNs to create: 1

LUN Name

[✓ Name] Sybex-Test-RAID5

Automatically assign LUN IDs as LUN Names

How much parity is set aside for data protection, and the overall fault tolerance of the RAID set, depends on the RAID level employed. For example, RAID 5 can tolerate a single drive failure, whereas RAID 6 can tolerate two drive failures. We will discuss these in more detail shortly.

Figure 4.6 shows a single drive in a five-drive RAID set being reserved for parity data. In this example, the usable capacity of the RAID set is reduced by 20 percent. This RAID set can also tolerate a single drive failure.

FIGURE 4.6 Five-drive RAID set with a single parity drive

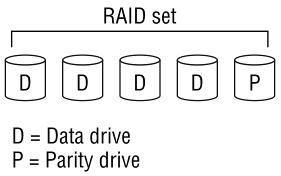
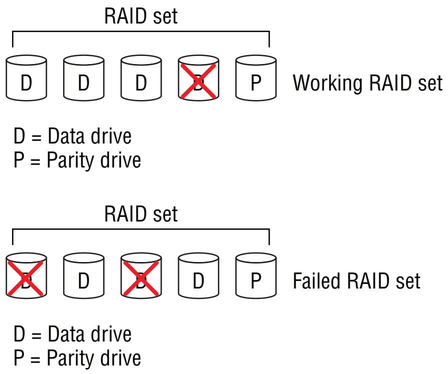


Figure 4.7 shows the same five-drive RAID set after experiencing single and double drive failures.

FIGURE 4.7 Five-drive RAID set with single and double drive failures



If the RAID set in Figure 4.7 had contained two parity drives (higher fault tolerance), it would have survived the double drive failure scenario. However, the second parity drive would have reduced the usable capacity of the RAID set from 80 percent to 60 percent.

Parity is usually computed via *exclusive OR (XOR)* calculations, and it doesn't really matter whether your RAID controller works with odd or even parity—so long as it works it out properly.

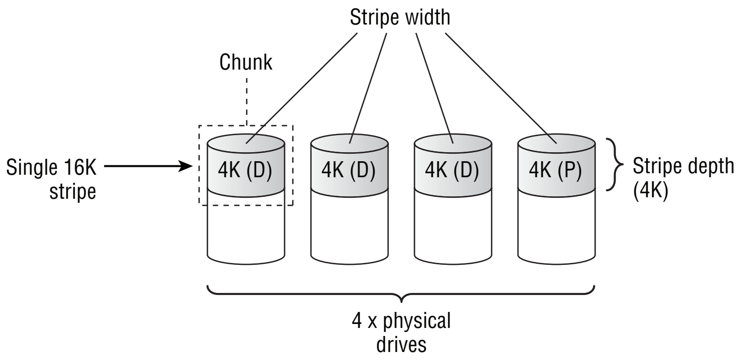
Parity-based RAID schemes can suffer performance impact when subject to high write workloads, especially high write workloads composed of small-block random writes. This is known as the *write penalty*, which we discuss in detail later. When it comes to high read workloads, parity-based RAID schemes perform well.

RAID Stripes and Stripe Sizes

All forms of traditional RAID that employ striping and parity work on the basis of stripes. Stripes are made from a row of chunks. The size of these chunks determines the stripe depth and stripe size, while the number of members in the RAID set determines the stripe width. I think we need a diagram!

Figure 4.8 shows a four-drive RAID set configured as RAID 5 (3+1) and has explanations for each of the elements that make up the stripe.

FIGURE 4.8 Stripe elements



Now for a few quick principles and a bit of lingo that apply to most traditional RAID sets.

Stripes are made up of chunks from each drive in the RAID set. Each *chunk* is made from contiguous blocks in a single physical drive. All chunks in a RAID configuration are of equal size—usually a multiple of the drive's sector size—and this size is known as the *stripe depth*. *Stripe width* is equal to the number of drives in the RAID set. The RAID set in Figure 4.8 has the following characteristics:

- Stripe depth is 4 KB. This is because each chunk is 4 KB, and the chunk size is equal to the stripe depth.
- Stripe size is 16 KB. Four 4 KB chunks.
- Stripe width is 4. There are four members in the RAID set.

When a host writes to a volume on a RAID controller, the RAID controller writes this data out in stripes so that it touches all drives in the RAID set. For example, if a host writes 24 KB to the RAID set in Figure 4.8, the RAID controller will write this out as two *stripes*. This is because each stripe has 12 KB of usable data and 4 KB of parity. As it writes this data out in stripes, it writes 4 KB to drive 0, before switching to drive 1, where it writes the next 4 KB, and so on. Writing 24 KB to this RAID set will require two *rows*, or stripes.

NOTE Storage arrays—themselves RAID controllers—tend to have far larger stripe sizes than host-based RAID controllers. It is not uncommon for a storage array to have a chunk size of 64 KB, 128 KB, or 256 KB. If the storage array has a chunk size of 256 KB, then a RAID 5 (3+1) would have a stripe size of 1,024 KB. The larger stripe sizes seen in storage arrays tend to be aligned to the slot size of the array's cache.

Chunk Size and Performance

The following can be used as a guiding principle that will work in most situations: If your I/O size will be large, you want to go with a smaller chunk size. If your I/O size will be small, you want larger chunks.

So, what constitutes a small or a large chunk size? Generally, small chunks can range from as small as 512 bytes to maybe 4 KB or 8 KB. Anything larger than that is starting to get into the realms of larger chunks.

Small I/O and Large Chunks

Small I/Os, such as 4 KB I/Os common in database environments, tend to work a lot better with larger chunk sizes. The objective here is to have each drive in the RAID set busy working and seeking independently on separate I/Os—a technique known as *overlapping I/O*.

If you have a larger chunk size with small I/O, you can easily incur increased positional latency. That is, while the heads on one drive might have arrived in position and be ready to service its portion of the I/O, the heads on the other drives in the RAID set might not yet have arrived in position, meaning that you have to wait until the heads on all drives in the RAID set arrive in place before the I/O can be returned. And if you subject your RAID controller to lots of small-block random I/O with a large chunk size, this can take its toll and reduce performance.

Large I/O and Small Chunks

Low I/O environments that work with large files often benefit from a smaller stripe size, as this allows each large file to span all physical drives in the RAID set. This approach enables each file to be accessed far more quickly, involving every drive in the RAID set to read or write the file. This configuration is not conducive to overlapping I/O, as all drives in the array will be used for reading or writing the single large file.

Common use cases for small chunk sizes include applications that access large audio/video files or scientific imaging applications that often manipulate large files.

Mixed Workloads

Fortunately, the default setting on most RAID controllers is usually good enough and designed for mixed workloads. It is recommended to change the defaults only if you know what you are doing and know what your workload characteristics will be. If you don't know either, don't mess with the chunk and stripe size. Also most storage arrays don't let you fiddle with this, as it is tightly aligned to cache slot size and other array internals.

Hot-Spares and Rebuilding

Some RAID arrays contain a spare drive that is referred to as a *hot-spare* or an *online spare*. This hot-spare operates in standby mode—usually powered on but not in use—during normal operating circumstances, but it is automatically brought into action, by the RAID controller, in the event of a failed drive. For example, a system with six drives could be configured as RAID 5 (4+1) with the remaining drive used as a hot-spare.

The major principle behind having hot-spares is to enable RAID sets to start rebuilding as soon as possible. For example, say your RAID set encounters a failed drive at 2 a.m., and nobody is on site to replace the failed drive. If it has a hot-spare, it can automatically start the rebuild process and be back on the road to recovery by the time you arrive at work in the morning. Obviously, larger disk drives take longer to rebuild, and in today's world, where drives are in excess of 4 TB, these could take days to rebuild.

The process of rebuilding to a hot-spare drive is often referred to as *sparing out*.

Hot-spares are not so common in host-based RAID controllers but are practically mandatory in storage arrays, with large storage arrays often containing multiple hot-spare drives.

Distributed Sparing and Spare Space

Some modern storage arrays don't actually contain physical hot-spare drives. Instead, they reserve a small amount of space on each drive in the array and set this space aside to be used in the event of drive failures. This is sometimes referred to as *distributed sparing.* For example, a storage array with 100 1 TB drives might reserve ~2 GB of space on each drive as *spare space* to be used in the event of drive failures. This space will be deducted from the overall usable capacity of the array.

This amount of reserved *spare space* is the equivalent of two 1 TB drives, meaning the array has the equivalent of two spare drives. In the event of a drive failure, the array will rebuild the contents of the failed drive to the *spare space* on every surviving drive in the array. This approach has the advantage of knowing that the drives that the spare space are on are in working condition. Having dedicated spare drives that are used only in the event of drive failures is prone to the hot-spare itself failing when needed most. Also of significance is that all the drives in the system are available all the time, increasing system performance and allowing RAID rebuilds to be distributed across all drives, making rebuild operations faster and lowering the performance impact.

Of course, this example was hugely oversimplified, but serves to explain the principle.

Rebuilding tends to be accomplished by one of two methods:

Drive copy, used when rebuilding in mirrored sets such as RAID 1

Correction copy/parity rebuild, used when rebuilding sets based on parity protection

Performance Impact of Rebuilds

Drive copy rebuilds are computationally simple compared to parity rebuilds, meaning they are far faster than having to reconstruct from parity.

RAID 1 mirror sets always recover with a drive copy, but parity-based RAID schemes such as RAID 5 and RAID 6 usually have to recover data via a parity rebuild.

Let's look at drive copy rebuilds and parity rebuilds:

- Drive copy operations are the preferred rebuild method but can be performed only if a drive is proactively failed by the RAID controller. This means that the drive has not actually failed yet, but the drive thresholds that the RAID controller monitors indicate that a failure is about to occur. This is known as a *predictive failure*. Drive copy rebuilds place additional stress on only the failing drive and the hot-spare.
- Parity rebuilds are the absolute last line of defense once a drive in a parity set has actually failed. These are computationally more intensive than drive copies and place additional load on all drives in the RAID set including the hot-spare—all surviving drives are read from stripe 0 through to the last stripe, and for each stripe, the missing data is reconstructed from by parity via exclusive OR (XOR) calculations. This additional load on the surviving drives in the RAID set is sometimes blamed when a second drive fails during the parity rebuild.

This point about additional load being blamed on second drive failures is potentially significant. During a rebuild operation, additional load is placed on all drives in the RAID set. It is entirely possible that this additional load can cause other drives in the RAID set to fail. This risk is being addressed by some of the more modern RAID implementations.

RAID Rebuild Priorities

Also, many RAID controllers, including storage arrays, allow you to prioritize the rebuild process. Giving rebuilds a lower priority leads to longer rebuilds, whereas giving rebuilds a high priority can have an impact on user I/O for the duration of the rebuild operation.

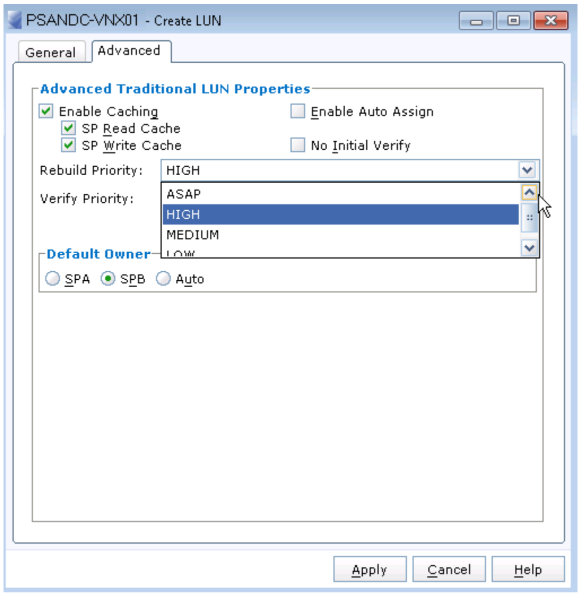
If you have large pools or single-parity RAID, you may wish to increase the priority of the rebuild so that your window of exposure to a second drive failure is decreased. In contrast, if you employ dual-parity RAID, you may feel more comfortable lowering the priority of the rebuild in the knowledge that you can safely suffer a second drive failure without losing data.

Figure 4.9 shows a screenshot of an EMC CLARiiON array showing available options for rebuild priority.

Rebuild Domains and Blast Zones

When rebuilding a failed drive in a RAID set, the RAID set is usually the *rebuild domain*. This means that all the rebuild activity and overhead is placed on the drives within the RAID set. Similarly, the *blast zone* refers to the area of impact when data is lost from a failed RAID set (such as a double drive failure in a RAID 5 set). When using traditional RAID sets, the blast zone is just the failed RAID set and any volumes using that RAID set. However, in more-modern arrays that implement storage pools comprising multiple RAID sets, a failed RAID set can cause data loss on every volume in the pool that the RAID set is associated with, dramatically increasing the size of the blast zone.

FIGURE 4.9 Setting the rebuild priority on an EMC array


PSANDC-VNX01 - Create LUN
General
Advanced
Advanced Traditional LUN Properties
Apply
Cancel
Help

Enable Caching [✓]
SP Read Cache [✓]
SP Write Cache [✓]
Enable Auto Assign [ ]
No Initial Verify [ ]
Rebuild Priority: HIGH
Verify Priority:
ASAP
HIGH
MEDIUM
LOW
Default Owner: SPA [✓ SPB] Auto

RAID Controllers

A RAID controller can be either software or hardware. Software RAID controllers are exactly that, software, and are often implemented via a logical volume manager. Hardware RAID controllers, on the other hand, are dedicated hardware designed specifically for performing RAID and volume management functions.

Hardware RAID controllers come in two major flavors:

- Internal
- External

As previously mentioned, hardware RAID controllers are dedicated hardware, either on the motherboard or as a PCIe card, that offload all RAID functions from the host CPU and RAM. These RAID functions include creating logical disks (referred to in SCSI parlance as logical units and usually abbreviated as *LUN*), striping, mirroring, parity calculation, rebuilding, hot-spares, and caching.

Write-Back Caching on Internal RAID Controllers

Most internal RAID controllers have an onboard DRAM/NVRAM cache used to speed up reads and writes. However, in order to safely speed up writes, a RAID controller must be able to protect its cache from losing data in the event of a low of power. There are two common approaches to this with modern RAID controllers:

Battery-Backed Caching In battery-backed caching, a small battery or capacitor (SuperCap) is placed on the RAID controller card and used to maintain power to the DRAM cache so that its contents are not lost when the power fails. However, power must be restored before the battery runs out. Otherwise, the data will still be lost.

Flash-Backed Caching In flash-backed caching, a battery or SuperCap is placed on the RAID controller card and used to power the DRAM cache long enough to copy its contents to flash memory so that its contents are preserved.

Battery-backed write-back caching (BBWC) is losing popularity in the wake of flash-backed caches. However, they still exist.

Caching technology is like oxygen to the storage industry. Starve a storage solution of cache, and its performance will drop like a stone. The benefits of placing a cache in front of spinning disks is covered in several chapters throughout this book, but caching also has a positive benefit in RAID systems. Any RAID controller, or storage array, that caches all inbound writes should be able to avoid the parity overhead of most writes under normal operating circumstances. This is done by holding small writes in cache and coalescing them into larger full-stripe writes (FSWs), where the overhead for parity computation is negligible.

RAID Levels

This section covers the most common RAID levels seen in the real world: RAID 0, RAID 1, RAID 10, RAID 5, and RAID 6. We'll also explore the less commonly used RAID levels.

RAID 0

First and foremost, there is nothing *redundant* about RAID 0! This means it provides no protection for your data. No parity. No mirroring. Nothing. Nada!

If I could recommend one thing about RAID 0, it would be never to use it. While some folks might come up with niche configurations where it might have a place, I'd far rather play it safe and avoid it like the plague.

However, in case it comes up on an exam or a TV quiz show, we'll give it a few lines.

RAID 0 is striping without parity. The striping means that data is spread over all the drives in the RAID set, yielding parallelism. More disks = more performance. Also, the lack of parity means that none of the capacity is lost to parity. But that is not a good thing. The lack of parity makes RAID 0 a toxic technology that is certain to burn you if you use it.

RAID 0 Cost and Overhead

There is no RAID overhead with RAID 0, as no capacity is utilized for mirroring or parity. There is also no performance overhead, as there are no parity calculations to perform. So, RAID 0 is nothing but good news on the cost and performance front. However, it is a ticking time bomb on the resiliency front.

Use Cases for RAID 0

Potential use cases for RAID 0 include the following:

1. If you want to lose data
2. If you want to lose your job
3. If your data is 100 percent scratch, losing it is not a problem, and you need as much capacity out of your drives as possible
4. As long as there is some other form of data protection such as network RAID or a replica copy that can be used in the event that you lose data in your RAID 0 set
5. If you want to create a striped logical volume on top of volumes that are already RAID protected

Options 1 and 2 are *not* recommended.

Option 3 is a potential poisoned chalice. Far too many systems start out their existence being *only for testing purposes* and then suddenly find that they are mission critical. And you do not want anything mission critical built on the sandy foundation of RAID 0.

Option 4 is more plausible but could also end up being a poisoned chalice. When data is lost, people always look for somebody else to blame. You could find the finger of blame being pointed at you.

Option 5 is quite common in the Linux world, where the LVM volume manager is well-known and very popular. However, it is absolutely vital that the underlying volume supporting the RAID 0 striped volume is RAID protected. Exercise 4.1 walks you through an example of creating a striped volume (RAID 0) with Linux LVM.

EXERCISE 4.1

Creating a Striped Volume with Linux LVM

Assume that you have a Linux host with four 100 GB volumes that are RAID 6 volumes: `sda`, `sdb`, `sdc`, and `sdd`. You will create a single striped volume (RAID 0) on top of these four RAID 6 volumes. Follow these steps:

1. Run the `pvcreate` command on all four of the RAID 6 volumes. Be careful, because doing this will destroy any data already on these volumes.

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
```

```
  # pvcreate /dev/sdc
  # pvcreate /dev/sdd
```

2. Now you'll create a volume group called `vg_sybex_striped` using your four volumes that you just ran the `pvcreate` commands against.

```
  # vgcreate vg_sybex_striped /dev/sda /dev/sdb /dev/sdc /dev/sdd
```

3. Once the volume group is created, run a `vgdisplay` command to make sure that the volume group configuration has been applied and looks correct.

```
  # vgdisplay
  --- Volume Group ---
VG Name              vg_sybex_striped
VG Access            read/write
VG Status            available/resizable
VG #                 2
<output truncated>
```

4. Once you have confirmed that the volume group configuration is as expected, run the following `lvcreate` command to create a single 200 GB logical volume.

```
  # lvcreate -i4 -I4 -L 200G -n lv_sybex_test_vol vg_sybex_striped
```

The `-i` option ensures that the logical volume will use all four physical volumes you created with the `pvcreate` command. The `-I` option specifies a 4 KB stripe size. The `-L` command specifies a size of 200 GB.

You now have a 200 GB RAID 0 (striped) logical volume that is sitting on top of four 100 GB RAID 6 volumes. These RAID 6 volumes are SAN-presented volumes, meaning that the RAID 6 protection is being provided by the storage array. Also, you have ~200 GB of free space in the volume group, which you can use to create more logical volumes or to expand your existing logical volume in the future.

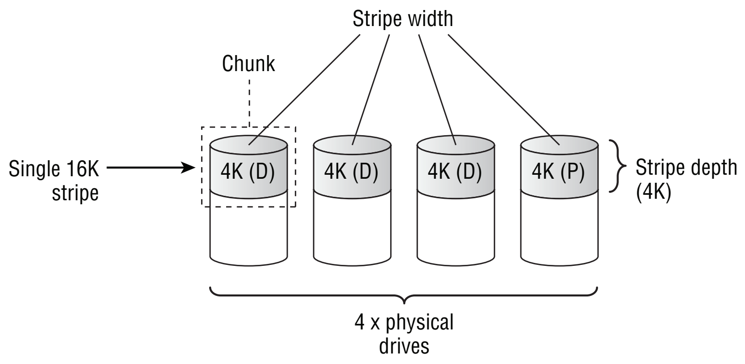
What Happens When a Drive Fails in a RAID 0 Set

You lose data.

End of story. You will possibly face the end of employment in your current role. You won't need to worry about performance degradation or rebuild times. Neither will be happening. If you are lucky, you will have a remote replica or backup to enable some form of recovery.

Figure 4.10 shows the career-limiting RAID 0.

FIGURE 4.10 The abomination of RAID 0



Real World Scenario

RAID 0 Does Not Protect Your Data!

A company lost data when a RAID 0 volume failed. This data had to be recovered from backup tapes based on a backup taken nearly 24 hours earlier. The administrator responsible for the storage array insisted he had not created any RAID 0 volumes and that there must be a problem with the array. The vendor analyzed logs from the array and proved that the administrator had indeed created several RAID 0 volumes. The logs even showed the dates on which the volumes were created and had recorded the administrator's responses to warnings that RAID 0 volumes would lose data if any drive failed. Needless to say, the administrator did not last long at the company.

RAID 1

RAID 1 is mirroring. A typical RAID 1 set comprises two disks—one a copy of the other. These two disks are often referred to as a *mirrored pair*. As a write comes into the RAID controller, it is written to both disks in the mirrored pair. At all times, both disks in the mirror are exactly up-to-date and in sync with each other. All writes to a RAID 1 volume are written to both disks in the RAID set. If this is not possible, the RAID set is marked as *degraded*.

NOTE *Degraded mode* refers to the state of a RAID set when one or more members (drives) of the set is failed. The term *degraded* refers to both degraded redundancy and degraded performance. In degraded mode, a RAID set will continue to service I/O; it will just be at greater risk of losing data should another drive fail, and performance may be impeded by data having to be reconstructed from parity. The latter is not the case for RAID 1, which does not use parity. However, even in a RAID 1 set, read performance may be degraded as the read can no longer be serviced from either disk in the mirrored pair.

A RAID 1 set contains a minimum of two drives. As the term *mirroring* suggests, one drive is a mirror copy of the other. This configuration is often written as *RAID 1 (1+1)*. The *1+1* refers to one drive for the primary copy and one drive for the mirrored copy. If either of the drives in the RAID set fails, the other can be used to service both reads and writes. If both drives fail, the data in the RAID set will be lost.

RAID 1 is considered a high-performance and safe RAID level. It's considered *high performance* because the act of mirroring writes is computationally lightweight when compared with parity-based RAID. It is considered *safe* because rebuild times can be relatively fast. This is because a full copy of the data still exists on the surviving member, and the RAID set can be rebuilt via a fast *drive copy* rather than a slower *parity rebuild*.

All good.

Exercise 4.2 shows how to create a mirrored logical volume with Linux LVM.

EXERCISE 4.2

Creating a Mirrored Logical Volume with Linux LVM

Let's assume you have a Linux host that is already presented with two 250 GB volumes that are not RAID protected. These could be SAN volumes or locally attached DAS drives: `sda` and `sdb`. Now let's form these two volumes into a mirrored logical volume (RAID 1).

1. Run the `pvcreate` command on both of the 250 GB volumes (`sda` and `sdb`). This is a destructive process that will destroy any data already on the devices.

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
```

2. Now create a volume group called `vg_sybex_mirrored` by using your two volumes that you just ran the `pvcreate` commands against.

```
# vgcreate vg_sybex_mirrored /dev/sda /dev/sdb
```

3. Once the volume group is created, run a `vgdisplay` command to make sure that the volume group configuration has been applied and looks correct.

```
  # vgdisplay
  --- Volume Group ---
VG Name               vg_sybex_mirror
VG Access             read/write
VG Status             available/resizable
VG #                  1
<output truncated>
```

4. Once it is confirmed that the volume group configuration is as expected, run the following `lvcreate` command to create a single 200 GB mirrored logical volume.

```
  # lvcreate -L 200G -m1 -n lv_sybex_mirror_vol vg_sybex_mirrored
```

The `-L` command specifies a size of 200 GB. It is worth noting that the 200 GB mirrored logical volume will not consume all the capacity of the volume group you created. This is fine.

You now have a 200 GB RAID 1 (mirrored) logical volume that is sitting on top of two 250 GB devices.

RAID 1 Cost and Overhead

The downside to RAID 1—and it's only a downside to the storage administrator and the cost center owner who owns the storage budget—is that RAID 1 sacrifices a lot of capacity on the altar of protection. All RAID 1 configurations use 50 percent of total physical capacity to provide the protective mirror. This sacrificed capacity is referred to as *RAID overhead*.

Also, say we assume a capital expenditure (cap-ex) cost of $5,000 per terabyte and you need to purchase 32 TB. On the face of it, this might sound like $160,000. However, if this needs to be RAID 1 protected, you need to buy double the raw capacity, which will obviously double the cap-ex cost from $160,000 to $320,000. That hurts!

For this reason, RAID 1 tends to be used sparingly—no pun intended.

Figure 4.11 shows two 600 GB drives in a RAID 1 set. Although the resulting logical drive is only 600 GB, it is actually consuming all the available space in the RAID set because this is a RAID 1 set and therefore loses half of its physical raw capacity to RAID overhead.

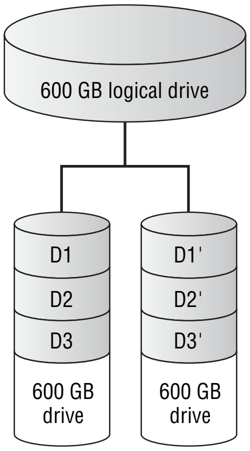
Use Cases for RAID 1

If you don't keep a tight rein on it, everybody will be asking for RAID 1. Database Administrators (DBAs) are especially guilty of this, as they have all been indoctrinated with anti–RAID 5 propaganda. If they insist on RAID 1, ask them to put their money where their mouth is and cross-charge them.

RAID 1 is an all-around good performer but excels in the following areas:

- Random write performance. This is because there is no write penalty for small random writes as there is with RAID 5 and RAID 6 (more on this later).
- Read performance. Depending on how your RAID controller implements RAID 1, read requests can be satisfied from either drive in the mirrored pair. This means that for read performance, you have all of the IOPS from both drives. Not all RAID controllers implement this.

FIGURE 4.11 RAID 1 mirror



What Happens When a Drive Fails

When a drive fails in a RAID 1 set, the set is marked as degraded, but all read/write operations continue. Read performance may be impacted.

Once the failed drive is replaced with a new drive or a hot-spare kicks in, the RAID set will be rebuilt via a drive copy operation whereby all the contents of the surviving drive are copied to the new drive. Once this drive copy operation completes, the RAID set is said to be back to *full redundancy*.

RAID 10

RAID 1+0, often referred to as *RAID 10*, and pronounced *RAID ten*, is a hybrid of RAID 1 (mirror sets) and RAID 0 (stripe sets). The aim is to achieve the best of both worlds. And generally speaking, it works exactly like that.

Let's look at how a RAID 10 set is created. Assume we have four 600 GB drives and we want to create a RAID 10 set. We take the first two drives and create a mirrored pair from them. Then we take the last two drives and make a separate mirrored pair from them.

This gives us two logical 600 GB drives, each of which is RAID 1 protected. The next step is then to take these two 600 GB RAID 1–protected drives and create a stripe set across them. The stripe set doesn't add any mirroring or parity, so no additional capacity is lost. All it does is increase performance by striping data over more drives. The net result now is 1,200 GB of capacity that is both mirrored and striped.

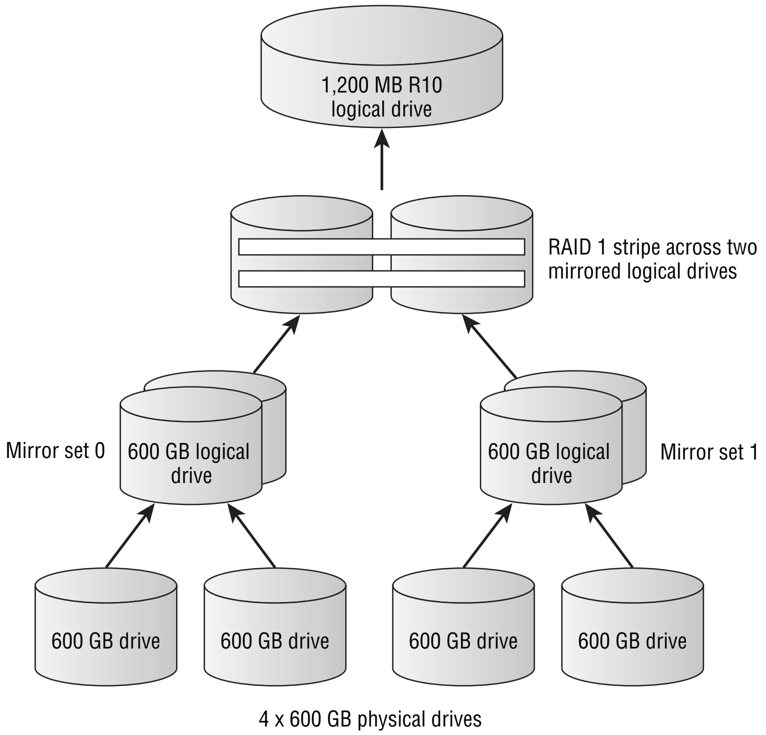
The Point of RAID 10

The thinking behind RAID 10 is pretty straightforward: to get the protection and performance of RAID 1 and turbo boost it with an additional layer of striping. That striping adds more drives to the RAID set, and drives = IOPS and MB/sec!

So, as long as you can afford the capacity overhead that comes with any derivative of RAID 1, then RAID 10 potentially offers the best mix of performance and protection available from the traditional RAID levels.

Figure 4.12 shows graphically how a RAID 10 set is created, and Exercise 4.3 walks us through the high level process of creating one.

FIGURE 4.12 Creating a RAID 10 set



EXERCISE 4.3

Creating a RAID 10 RAID Set

Your RAID controller will do this for you, so this is purely conceptual. Let's assume you have four 1 TB physical drives in your server/storage array. To create a RAID 10 set, follow these steps:

1. Use two of the drives to create a 1 TB mirrored volume.
2. Do exactly the same with the remaining two drives. At the end of these two steps, you would have two 1 TB mirrored volumes.
3. Take these two mirrored volumes and create a single 2 TB RAID 0 striped volume across them.

The end result would be a single 2 TB volume that is protected with RAID 1 and has the added performance benefit of all reads and writes being striped across the drives.

Beware of RAID 01

RAID 10 and RAID 01 are not the same, and people often get the two mixed up. You always want RAID 10 rather than RAID 01. The technical difference is that RAID 01 first creates two stripe sets and then creates a mirror between them. The major concern with RAID 01 is that it is more prone to data loss than RAID 10. Some RAID controllers will automatically build a RAID 10 set even if they call it RAID 1.

RAID 5

Despite its critics, *RAID 5* probably remains the most commonly deployed RAID level—although most people consider it a best practice to use RAID 6 for all drives that are 1 TB or larger.

RAID 5 is known technically as *block-level striping with distributed parity*. This tells us two things:

Block level tells us that it is not bit or byte level. Block size is arbitrary and maps to the chunk size we explained when discussing striping.

Distributed parity tells us that there is no single drive in the RAID set designated as a *parity drive*. Instead, parity is spread over all the drives in the RAID set.

RAID 5 Cost and Overhead

Even though there is no dedicated parity drive in a RAID 5 set, RAID 5 always reserves the equivalent of one disk worth of blocks, across all drives in the RAID set, for parity.

This enables a RAID 5 set—no matter how many members are in the set—to suffer a single drive loss without losing data. However, if your RAID 5 set loses a second drive before the first is rebuilt, the RAID set will fail and you will lose data. For this reason, you will want your RAID 5 sets to have fast rebuild times.

Common RAID 5 configurations include the following:

- RAID 5 (3+1)
- RAID 5 (7+1)

RAID Notation

Let's take a quick look at common RAID notation. When writing out a RAID set configuration, the following notation is commonly used:

RAID X (D+P)

X refers to the RAID level, such as 1, 5, or 6.

D indicates the number of data blocks per stripe.

P indicates the number of parity blocks per stripe.

So if our RAID set has the equivalent of three data drives and one parity drive, this would make it a RAID 5 set, and we would write it as follows: *RAID 5 (3+1).*

As the rules of RAID 5 state only that it is distributed block-level parity, it is possible to have RAID 5 sets containing a lot more drives than the preceding examples—for example, RAID 5 (14+1). However, despite the number of drives in a RAID 5 set, as always, only a single drive failure can be tolerated without losing data.

The amount of capacity lost to parity in a RAID 5 set depends on how many data blocks there are per parity stripe. The calculation is parity/data × 100 expressed as a percentage.

So, in our RAID 5 (3+1) set, we lose 25 percent of capacity to parity, as 1/4 × 100 = 25. A RAID 5 (7+1) set is more space efficient in that it sacrifices only 12.5 percent of total space to parity: 1/8 × 100 = 12.5. And at a far extreme, RAID 5 (15+1) sacrifices only 6.25 percent of space to parity.

Be careful, though. While RAID 5 (15+1) yields a lot more usable space than RAID 5 (3+1), there are two drawbacks:

- Longer rebuilds. Rebuild times for the 15+1 RAID set will be longer than the 3+1 set because the XOR calculation will take longer, and there are more drives to read when reconstructing the data.
- Lower performance, because it's harder to create FSWs and avoid the RAID 5 write penalty. We will discuss this shortly.

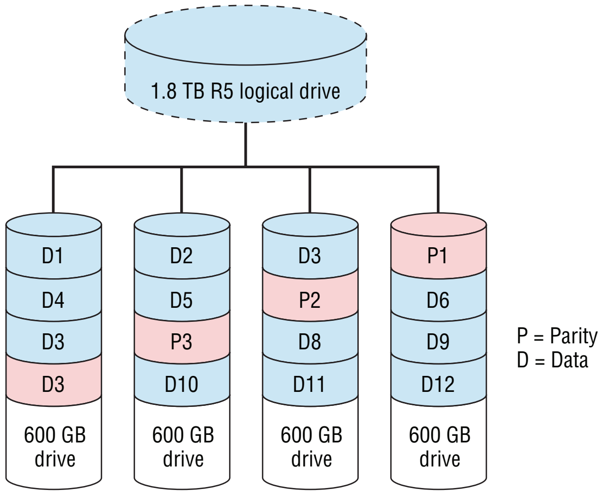
What Happens When a Drive Fails in a RAID 5 Set

When a drive fails in a RAID 5 set, you have not lost any data. However, you need the RAID set to rebuild as quickly as possible, because while there is a failed drive in your RAID set, it is operating in *degraded mode*. The performance of the RAID set and its ability to sustain drive failure are degraded. So it's a good idea to get out of degraded mode as quickly as possible.

Assuming there is a hot-spare available, the RAID controller can immediately start rebuilding the set. This is achieved by performing exclusive OR (XOR) operations against the surviving data in the set. The XOR operation effectively tells the RAID controller what was on the failed drive, enabling the hot-spare drive to be populated with what was on the failed drive. Once the rebuild completes, the RAID set is no longer in degraded mode, and you can relax.

Figure 4.13 shows a RAID 5 (3+1) array.

FIGURE 4.13 RAID 5 (3+1) array with rotating parity



RAID 5 Write Penalty

Parity-based RAID schemes, such as RAID 5 and RAID 6, suffer from an inherent performance defect known as the *write penalty*. This is especially a problem in situations where there is high write activity, particularly high random write activity. It occurs because calculating parity for small-block writes incurs additional I/O on the backend as follows. A small-block random write to a RAID 5 group incurs four I/Os for each write:

1. Read old data.
2. Calculate new data (in memory).

3. Write new data.
4. Calculate and write new parity.

Let's look at a quick example. Figure 4.13 shows a simplified RAID 5 (3+1) RAID set. Let's assume we wanted to write only to block D1 on row 0. To calculate the new parity for row 0, the RAID controller will have to read the existing data, read the existing parity, write the new data to D1, and write the new parity to P1. This overhead is a real pain in the neck and can trash the performance of the RAID set if you subject it to high volumes of small-block writes. For this reason, many application owners, especially DBAs, have an aversion to RAID 5. However, many RAID 5 implementations go a long way in attempting to mitigate this issue.

Full Stripe Writes Do Not Suffer from the Write Penalty

Because the write penalty applies to only small-block writes, where the write size is smaller than the stripe size of the RAID set, performing larger writes avoids this problem. This can be achieved via caching.

By putting a large-enough cache in front of a RAID controller, the RAID controller is able store up small writes and bundle them together into larger writes. This is a technique called *write coalescing*. The aim is to always write a full stripe—because writing a full stripe does not require additional operations to read blocks that you are not writing to, and you can write the *data* and *parity* in a single operation.

However, write coalescing is a feature of high-end external RAID controllers (storage arrays) and is an advanced technique. It is not usually a feature of a hardware RAID controller in a server, and certainly not a technique used by software RAID controllers. The effectiveness of write coalescing can also be diminished if the RAID controller is subject to particularly high write workloads.

Distributed RAID Schemes May Limit the Impact of the Write Penalty

Distributed RAID refers to the ability to spread a RAID-protected volume over hundreds of drives, with each successive stripe existing on a separate set of drives.

Figure 4.14 shows a traditional four-drive RAID group. In this figure, four consecutive stripes in a logical volume are shown.

FIGURE 4.14 Traditional RAID group with four consecutive stripes

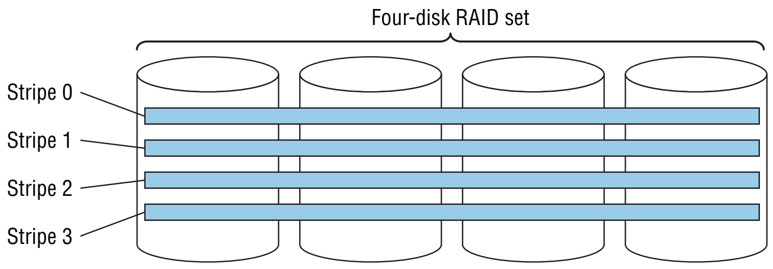
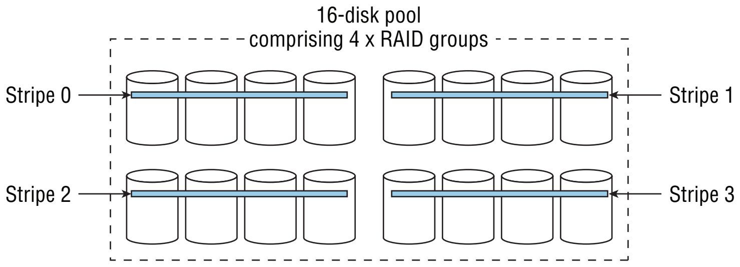


Figure 4.15 shows the same four stripes for the same logical volume, but this time spread across all 16 drives in a 16-drive pool.

FIGURE 4.15 Parallel RAID configuration with four consecutive stripes



While the configuration in Figure 4.15 doesn't avoid the write penalty, it can offset some of the impact by using all 16 drives in parallel and potentially writing multiple stripes in parallel.

NOTE The write penalty affects only parity-based RAID systems and does not afflict RAID 1 or RAID 0. It is most noticeable when experiencing high-write workloads, particularly high-random-write workloads with a small block-size.

RAID 6

RAID 6 is similar to RAID 5. Both do block-level striping with distributed parity. But instead of single parity, RAID 6 employs double parity. This use of double parity brings some clear-cut pros and cons.

NOTE RAID 6 is based on polynomial erasure codes. You will look at erasure codes later in the chapter as they may prove to be instrumental in the future of RAID and data protection, especially in cloud-based architectures.

RAID 6 Cost and Overhead

As with RAID 5, there is no dedicated parity drive in a RAID 6 set. The parity is distributed across all members of the RAID set. However, whereas RAID 5 always reserves the equivalent of one drive's worth of blocks for parity, RAID 6 reserves two drives' worth of blocks for two discrete sets of parity. This enables a RAID 6 set—no matter how many members are in the set—to suffer two failed drives without losing data. This makes RAID 6 an extremely safe RAID level!

Common RAID 6 configurations include the following:

- RAID 6 (6+2)
- RAID 6 (14+2)

The same as with RAID 5, a RAID 6 set can have an arbitrary number of members, but it *always* has two sets of discrete parity! So RAID 6 (30+2) is a valid RAID 6 configuration, so long as your RAID controller supports it and you have enough drives in the set.

The amount of capacity lost to RAID overhead in a RAID 6 set abides by the same rules as RAID 5. You divide 2 by the total number of drives in the set and then multiply that number by 100, as shown here:

- RAID 6 (14+2): 2/16 × 100 = 12.5 percent capacity overhead. This is sometimes referred to as 87.5 percent efficient.
- RAID 6 (30+2): 2/32 × 100 = 6.25 percent capacity overhead, or 93.75 percent efficient.

As with RAID 5, the more drives in the set, the slower the rebuild and the lower the write performance. Remember, parity-based RAID schemes suffer the write penalty when performing small-block writes. These can be offset by using cache to coalesce writes and perform full-stripe writes. However, the larger the RAID set (actually the larger the stripe size), the harder it is to coalesce enough writes to form a full-stripe write.

When to Use RAID 6

On the positive side, RAID 6 is rightly considered extremely safe. It can suffer two drive failures in the set without losing data. This makes it an ideal choice for use with large drives or for use in pools with large numbers of drives.

Large drives tend to be slow drives—in terms of revolutions per minute (RPM)—and therefore take a lot longer to rebuild than smaller, faster drives. For this reason, the window of exposure to a second drive failure while you are performing a rebuild is greatly increased. RAID 6 gives you an extra safety net here by allowing you to suffer a second drive failure without losing data. Also if you are deploying pools with hundreds of drives, if you were to lose a RAID set making up that pool, you could punch a hole in every volume using the pool. That would mean data loss on every volume using the pool! So for these two use cases, RAID 6 is very much the de facto choice.

RAID 6 performs well in high-read situations.

When to Not Use RAID 6

The major downside of RAID 6 is performance, especially performance of small-block writes in an environment experiencing a high number of writes. Because RAID 6 doubles the amount of parity per RAID stripe, RAID 6 suffers more severely from the write penalty than RAID 5. The four I/Os per small-block write for a RAID 5 set becomes six I/Os per write for a RAID 6 set. Therefore, RAID 6 is not recommended for small-block-intensive I/O requirements, and in the vast majority of cases, it is not used for any heavy write-based workloads.

Another area where RAID 6 suffers is in usable capacity. This is because RAID 6 has two parity blocks per stripe. However, this can be offset by deploying RAID 6 in large RAID sets such as RAID 6 (6+2) or quite commonly RAID 6 (14+2). These lose 25 percent and 12.5 percent, respectively. Don't forget, though, the larger your stripe size, the harder it is for the RAID controller to coalesce writes in cache for full-stripe writes.

When all is said and done, RAID 6 is rightly viewed as a slow, but safe, RAID choice. It is almost always the RAID level of choice for RAID groups containing drives over 1 TB in size.

Real World Scenario

RAID 6: A Safe Pair of Hands

A company suffered more than 30 failed drives in a single storage array over a period of less than a week—due to a bug with a specific batch of disk drives from a particular vendor. However, this company was employing large drive pools and therefore had deployed RAID 6. Despite losing so many drives in such a short period of time, no triple drive failures occurred in a single RAID set making up the pool, so no data was lost!

Less Commonly Used RAID Levels

Some RAID levels are not as commonly used as the previously discussed RAID levels. Even so, it is important for you to understand what these RAID levels are. RAID 2, RAID 3, and RAID 4 are briefly introduced here.

RAID 2

RAID 2 is bit-level striping and is the only RAID that can recover from single-bit errors. As it is not used in the real world, this is as much as you will need to know about it.

RAID 3

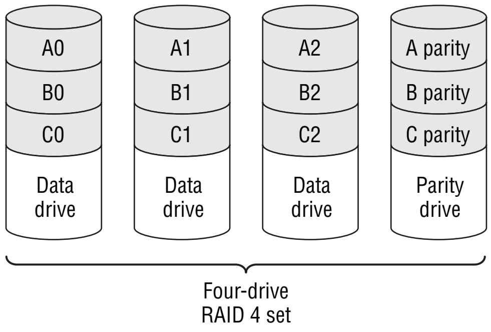
RAID 3 performs parity at the byte level and uses a dedicated parity drive. RAID 3 is extremely rare in the real world.

RAID 4

RAID 4 is similar to RAID 5. It performs block-level striping but has a dedicated parity drive.

One of the common issues with using dedicated parity drives is that the parity drive can become a bottleneck for write performance. Figure 4.16 shows a four-disk RAID 4 set with a dedicated parity drive. In this RAID configuration, each time you update just a single block on each row, you have to update the parity on the dedicated parity drive. This can cause the parity drive to become a bottleneck.

FIGURE 4.16 RAID 4 parity drive becoming a bottleneck



However, it has the advantage that the RAID set can be easily expanded by adding more disks to the set and requiring only the parity drive to be rebuilt. Other RAID schemes with distributed parity would require the entire RAID set to be rebuilt when adding drives.

RAID 4 is more common than RAID 2 and RAID 3 but is nowhere near as popular as RAID 5 and RAID 6.

NetApp's Data ONTAP supports RAID 4 and tightly integrates it with the Write Anywhere File Layout (WAFL) filesystem that is also part of Data ONTAP. By integrating the filesystem and the RAID, as long as the system is nearing capacity and being pushed to the limits, Data ONTAP is able to coalesce writes and avoid the performance impact of the dedicated parity drive (where the parity drive is hit harder than the other drives in the RAID set). Integrating filesystems and RAID schemes can have interesting benefits, which are covered at the end of the chapter when we discuss ZFS.

The Future of RAID

RAID technology has been around for decades. And it's fair to say that while most things in the technology world have moved on enormously in the past two decades, most implementations of RAID have not. Also, the disk drive bloat problem—drives getting bigger and bigger but not faster—is leading to longer and longer rebuild times. And longer rebuild times means you are exposed to the risks of additional drive failures for longer periods of time. This has led to the point we are at now, where traditional approaches to RAID just aren't up to the challenge of today's business and IT demands.

Fortunately, some interesting and important new RAID techniques and technologies have started cropping up. Some of the most popular include those that implement forms of parallel, or distributed, RAID. We will cover a few of these new approaches and technologies that highlight some of the advances made.

NOTE As we review some of these modern approaches to RAID, we will reference some vendor-specific technologies. As we do this, the objective isn't to promote or denounce any specific technology. Our interest is purely in looking academically at some of the ways they differ from the traditional approaches and how they attempt to address today's requirements.

IBM XIV and RAID-X

IBM XIV storage arrays implement a form of RAID 1 technology referred to as *RAID-X*. The reason we are interested in RAID-X is that it is not any kind of RAID 1 your grandmother will recognize. Yes, it loses 50 percent capacity to protection, and yes, it is susceptible to double drive failures. But this is 21st-century, turbo-charged, object-based, distributed RAID. So there are lots of cool things for us to look at.

Object-Based RAID

Object based means that RAID protection is based on objects rather than entire drives. XIV's implementation of object-based RAID is based on 1 MB extents that it refers to as *partitions*. Each volume on an XIV is made up of multiple 1 MB extents, and it is these 1 MB extents that are RAID-X protected.

However, as all volumes on an XIV are natively thin provisioned, extents are allocated to volumes on demand as data is written to volumes. Interestingly, this leads to improved reprotect and rebuild operations when a drive fails, which we will cover shortly.

Distributed RAID

The object-based nature of RAID-X enables the 1 MB extents that make up RAID-X volumes to be spread out across the entire backend of the XIV array—hundreds of drives. This wide-striping of volumes across the entire backend leads to massively parallel reads, writes, and reprotect operations. This is totally different from the two-drive RAID 1 configurations you looked at earlier!

RAID-X and Reprotecting

RAID-X is susceptible to double drive failures. If a drive in an XIV fails and then a second drive fails before the reprotect operation for the first failure completes, the system *will* lose data. And because every XIV volume is spread over pretty much every drive on the backend, the blast zone of a double drive failure is immense. It could punch a small hole in every volume on the system—oh, dear! However, and this is significant, RAID-X is highly optimized to reprotect itself blazingly fast! It does this via massively parallel reprotection operations. Let's take a look.

NOTE With traditional drive-based approaches to RAID, when a drive fails and the RAID set goes into degraded mode, the only way to get the RAID set back to fully protected mode is to rebuild the contents of the failed drive to another drive—usually to a hot-spare. Drive rebuild operations can take serious amounts of time in the world of multi-TB drives. We're talking *days* to rebuild large drives on some arrays. Hence the recommendation is to use RAID 6 with large, slow drives. Traditional drive-based RAID also rebuilds entire drives even if there is only a tiny amount of data on the drives. RAID-X is altogether more intelligent.

NOTE Another reason double protection is recommended for large drives is that the chances of hitting an unrecoverable read error while performing the rebuild are greatly increased. If you have only single protection for large drives, and during a rebuild you encounter an unrecoverable read error, this can be effectively the same as a second drive failure, and you will lose data. So at least double protection is highly recommended as drives get bigger and bigger.

RAID-X Massively Parallel Rebuilds

When a drive fails in an XIV, RAID-X does not attempt to rebuild the failed drive, at least not just yet. Instead it spends its energy in *reprotecting* the affected data. This is significant.

Let's assume we have a failed 2 TB drive that is currently 50 percent used. When this drive fails, it will leave behind 1 TB of data spread over the rest of the backend that is now unprotected. This unprotected 1 TB is still intact, but each of those 1,024 1 MB extents no longer has a mirrored copy elsewhere on the array. And because this 1 TB of unprotected data is spread across all drives on the backend, the failure of any other drive will cause some of these unprotected 1 MB extents to be lost. So, in order to reprotect these 1 MB extents as fast possible, the XIV reads each extent from all drives on the backend, and writes new mirror copies of them to other drives spread across the entire backend—all in parallel. Of course, RAID-X makes sure that the primary and mirror copy of any of the 1 MB extents is never on the same drive, and it goes further to ensure that they are also on separate nodes/shelves.

This is an excellent example of a *massively parallel* many-to-many operation that will complete in just a few short minutes. It is rare to see a drive rebuild operation on an XIV last more than 20–30 minutes, and often they complete much quicker. The speed of the reprotect operation is vital to the viability of RAID-X at such large scale, as it massively reduces the window of vulnerability to a second simultaneous drive failure.

Also, unlike traditional RAID, the parallel nature of a RAID-X rebuild means that it avoids the increased load stress that is typically placed on all disks in the RAID set during normal RAID rebuild operations. In most XIV configurations, each drive shoulders a tiny ~1 percent of the rebuild heavy lifting.

RAID-X and Object-Based Rebuilds

Also, because RAID-X is object based, it reprotects only actual data. In our example of a 2 TB drive that is 50 percent full, only 1 TB of data needs reprotecting and rebuilding. Non-object-based RAID schemes will often waste time rebuilding tracks that have no data on them. By reprotecting only data, RAID-X speeds up reprotect operations.

Once the data is reprotected, RAID-X will proceed to rebuild the failed drive. This rebuild is a many-to-one operation, with the new drive being the bottleneck.

RAID-X on IBM XIV isn't the only implementation of modern object-based parallel RAID, but it is a well-known and well-debated example.

On the topic of object-based rebuilds, other storage arrays such as HP 3PAR, Dell Compellent, and XIO also employ this approach. However, these technologies use object-based RAID with **parity** and **double parity** rather than XIV's form of object based **mirroring**. As the technologies that employ object-based approaches to RAID are all modern arrays, this is clearly the next step in RAID technology.

WARNING

RAID-X, as well as almost all pool-based backend designs, increases the potential blast zone when double drive failures occur. Because all volumes are spread over all (actually most) drives on the backend, a double drive failure has the potential to punch a small whole in most, if not all, volumes. This is something to beware of and is a major reason that most conventional arrays recommend RAID 6 when deploying large pools. The idea is to reduce risk, and RAID-X does this by its super-fast massively parallel reprotect operations.

ZFS and RAID-Z

RAID-Z is a parity-based RAID scheme that is tightly integrated with the ZFS filesystem. It offers single, double, and triple parity options and uses a dynamic stripe width. This dynamic, variable-sized stripe width is powerful, effectively enabling every RAID-Z write to be a full-stripe write—with the exception of small writes that are usually mirrored. This is brilliant from a performance perspective, meaning that RAID-Z rarely, if ever, suffers from the read-modify-write penalty that traditional parity-based RAID schemes are subject to when performing small-block writes.

RAID-Z volumes are able to perform full-stripe writes for all writes by combining variable stripe width with the redirect-on-write nature of the ZFS filesystem. *Redirect-on-write* means that data is never updated in place, even if you are updating existing data. Instead, data is always written to a fresh, new location—as a full-stripe write of variable size—and the pointer table is updated to reflect the new location of the updated data, and the old data is marked as no longer in use. This mode of operation results in excellent performance, but begins to tail off significantly when free space starts to get low.

Recovery with RAID-Z

RAID-Z rebuilds are significantly more complex than typical parity-based rebuilds where simple XOR calculations are performed against each RAID stripe. Because of the variable size of RAID-Z stripes, RAID-Z needs to query the ZFS filesystem to obtain information about the RAID-Z layout. This can cause longer rebuild times if the pool is near capacity or busy. On a positive note, because RAID-Z and the ZFS filesystem talk to each other, rebuild operations rebuild only actual data and do not waste time rebuilding empty blocks.

Also, because of the integration of RAID-Z and ZFS, data is protected from silent corruption. Each time data is read, it is passed through a checksum (that was created the last time the data was updated) and reconstructed if the validation fails. This is something that block-based RAID schemes without filesystem knowledge cannot do.

RAID-TM

RAID-TM is a triple-mirror-based RAID. Instead of keeping two copies of data as in RAID 1 mirror sets, RAID-TM keeps three copies. As such, it loses 2/3 of capacity to protection but provides good performance and excellent protection if you can get past the initial shock of being able to use only 1/3 of the capacity you purchased.

RAID 7

The idea behind *RAID 7* is to take RAID 6 one step further by adding a third set of parity. At first glance, this seems an abomination, but its inception is borne out of a genuine need to protect data on increasingly larger and larger drives. For example, it won't be long before drives are so large and slow that it will take a full day to read them from start to finish, never mind rebuild them from parity! And as we keep stressing, longer rebuilds lead to larger windows of exposure, as well as the likelihood of encountering an unrecoverable read error during the rebuild.

RAIN

RAIN is an acronym for *redundant/reliable array of inexpensive nodes*. It is a form of network-based fault tolerance, where nodes are the basic unit of protection rather than drives or extents. RAIN-based approaches to fault tolerance are increasingly popular in scale-out filesystems, object-based storage, and cloud-based technologies. RAIN-based approaches to fault tolerance are covered in other chapters where we talk about scale-out architectures, object storage and cloud storage.

Erasure Codes

Erasure codes is another technology that is starting to muscle its way into the storage world. More properly known as *forward error correction (FEC)* or sometimes *information* *dispersal algorithms (IDA)*, this technology has been around for a long time and is widely used in the telco industry. Also, erasure codes are already used in RAID 6 technologies as well some consumer electronics technologies such as CDs and DVDs. And let's face it, any data protection and recovery technology that can still produce sounds from CDs after my children have had their hands on them must surely have something to offer the world of enterprise storage!

Whereas parity schemes tend to work well at the disk-drive level, erasure codes initially seem to be a better fit at the node level, protecting data across nodes rather than drives. They can, and already are, used to protect data across multiple nodes that are geographically dispersed. This makes them ideal for modern cloud-scale technologies.

Erasure codes also work slightly differently from parity. While parity separates the parity (correction codes) from the data, erasure codes expand the data blocks so that they include both real data and correction codes. Similar to RAID, erasure codes offer varying levels of protection, each of which has similar trade-offs between protection, performance, and usable capacity. As an example, the following protection levels might be offered using erasure code technology:

- 9/11
- 6/12

In the first example, the data can be reconstructed if only 9 out of the 11 data blocks remain, whereas 6/12 can reconstruct data if only 6 of 12 data blocks remain. 6/12 is obviously similar to RAID 1 in usable capacity.

While erasure codes are the foundation of RAID 6 and are finding their way into mass storage for backup and archive use cases, it is likely that they will become more and more integral to protection of *primary storage* in the enterprise storage world, and especially cloud-based solutions.

Finally, current implementations of erasure codes don't perform particularly well with small writes, meaning that if erasure codes do form the basis of data protection in the future, it is likely that we will see them used in cloud designs and maybe also in large scale-out filesystems, with more-traditional RAID approaches continuing to protect smaller-scale systems and systems that produce small writes, such as databases.

Chapter Essentials

The Importance of RAID RAID technologies remain integral to the smooth operation of data centers across the world. RAID can protect your data against drive failures and can improve performance.

Hardware or Software RAID Hardware RAID tends to be the choice that most people go for if they can afford it. It can offer increased performance, faster rebuilds, and hot-spares, and can protect OS boot volumes. However, software RAID tends to be more flexible and cheaper.

Stripe Size and Performance Most of the time, the default stripe size on your RAID controller will suffice—and often storage arrays will not let you change it. However, if you know your workloads will be predominantly small block, you should opt for a large stripe size. If you know your I/O size will be large, you should opt for a smaller stripe size.

RAID 1 Mirroring RAID 1 offers good performance but comes with a 50 percent RAID capacity overhead. It's a great choice for small-block random workloads and does not suffer from the write penalty.

RAID 5 RAID 5 is block-level interleaved parity, whereby a single parity block per stripe is rotated among all drives in the RAID set. RAID 5 can tolerate a single drive failure and suffers from the write penalty when performing small-block writes.

RAID 6 RAID 6 is block-level interleaved parity, whereby two discrete parity blocks per stripe are rotated among all drives in the RAID set. RAID 6 can tolerate a two-drive failure and suffers from the write penalty when performing small-block writes.

Summary

In this chapter we covered all of the common RAID level in use today, including those required for the CompTIA Storage+ exam. As part of our coverage, we looked at the various protection and performance characteristics that each RAID level provides, as well as cost and overheads for each. We also addressed some of the challenges that traditional RAID levels face in the modern IT world, and mentioned some potential futures for RAID.

Chapter

5

Fibre Channel SAN

TOPICS COVERED IN THIS CHAPTER:

- ✓ What a Fibre Channel SAN is
- ✓ HBAs and CNAs
- ✓ Switches and directors
- ✓ Cabling
- ✓ Fabric services
- ✓ Zoning
- ✓ SAN topologies
- ✓ FC routing and virtual fabrics
- ✓ Flow control
- ✓ Troubleshooting

This chapter gets under the hood with the important technologies that make up a Fibre Channel SAN. It covers the theory and the practical applications, and shares tips that will help you in the real world. You'll learn the basics about redundancy; cabling; switch, port, and HBA configuration; as well as more-advanced topics such as inter-switch links, virtual fabrics, and inter-VSAN routing.

By the end of the chapter, you will know all the necessary theory to deploy highly available and highly performing FC SANs, as well as a few of the common pitfalls to avoid. You'll also have the knowledge required to pass the FC SAN components of the exam.

What Is an FC SAN?

First up, let's get the lingo right. *FC SAN* is an abbreviation for *Fibre Channel storage area network*. Most people shorten it to SAN.

At the very highest level, an FC SAN is a storage networking technology that allows block storage resources to be shared over a dedicated high-speed Fibre Channel (FC) network. In a little more detail, *Fibre Channel Protocol (FCP)* is a mapping of the SCSI protocol over Fibre Channel networks. So in effect, SCSI commands and data blocks are wrapped up in FC frames and delivered over an FC network. There is plenty more to come about that in this chapter.

NOTE The word *channel* in the context of Fibre Channel is important. It comes from the fact that SCSI is a form of *channel technology*. Channel-based technologies are a bit different from normal network technologies. The goal of a channel is high performance and low overhead. Channel technologies also tend to be far less scalable than networking technologies. Networks, on the other hand, are lower performance, higher overhead, but massively more scalable architectures. FC is an attempt at networking a channel technology—aiming for the best of both worlds.

NOTE FC Protocol can run over optical (fiber) or copper cables, and 99 percent of the time it runs over optical fiber cables. Copper cabling in FC networks is usually restricted to black-box configurations or the cabling on the back-end of a storage array.

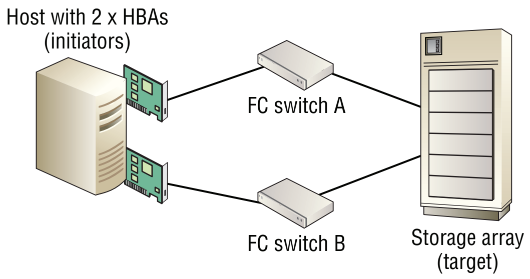
NOTE One final note just to confuse you: when referring to the Fibre Channel Protocol or a Fibre Channel SAN, *fibre* is spelled with *re* at the end. When referring to fiber cables, it is usually spelled with *er* at the end.

Because FCP encapsulates SCSI, it is technically possible to share any SCSI device over an FC SAN. However, 99.9 percent of devices shared on an FC SAN are disk storage devices, or tape drives and tape libraries.

Devices shared on an FC SAN, like all SCSI devices, are block devices. *Block devices* are effectively raw devices that for all intents and purposes appear to the host operating system and hypervisor as though they are locally attached devices. Also they do not have any higher levels of abstraction, such as filesystems, applied to them. That means that in an FC SAN environment, the addition of filesystems is the responsibility of the host accessing the block storage device.

Figure 5.1 shows a simple FC SAN.

FIGURE 5.1 Simple FC SAN high-level view



As shown in Figure 5.1, the following are some of the major components in an FC SAN:

- Initiators (commonly referred to simply as hosts or servers)
- Targets (disk arrays or tape libraries)
- FC switches

Initiators and targets in Figure 5.1 are SCSI concepts that can be thought of as *clients* and *servers* as in the client-server networking model. Initiators are synonymous with clients, whereas targets are synonymous with servers. More often than not, initiators issue read and write commands to targets over the FC SAN.

Initiators are usually hosts, or to be more accurate, they are HBA or CNA ports in a host. Targets are usually HBA or CNA ports in a storage array or tape library. And the SAN is one or more FC switches providing connectivity between the initiators and targets. Simple.

NOTE

HBA is an abbreviation for *host bus adapter*. *CNA* is an abbreviation for *converged network adapter*. Both are PCI devices that are installed in servers either as PCI expansion cards or directly on the motherboard of a server. When on an FC SAN, the drivers for these HBAs and CNAs make them appear to the host OS of the hypervisor as local SCSI cards. This means that any devices presented to the OS or hypervisor over the SAN via the HBA or CNA will appear to the OS or hypervisor as if it were a locally attached device. Storage arrays and tape devices on a SAN also have HBAs or CNAs installed in them.

Why FC SAN?

The traditional approach of deploying direct-attached storage (DAS) inside servers has pros and cons. Some of the pros are that DAS storage is accessed over very short, dedicated, low-latency, contention-free interconnects (SCSI bus). This guarantees quick and reliable access to devices. However, some of the cons of DAS include increased management, low utilization capacity islands, and limited number of devices such as disk drives.

The idea behind FC SAN technology was to keep the pros but overcome the cons of DAS storage. As such, SAN technology provides a high-speed, low-latency storage network (a channel) that is relatively contention free—especially when compared with traditional Ethernet networks. SANs also get rid of capacity islands by allowing large numbers of drives to be pooled and shared between multiple hosts. They also simplify management by reducing the number of management points. While FC SANs come at a cost ($), they have served the data center well for quite a number of years. However, they are under threat.

The emergence of several new and important technologies is threatening the supremacy of the SAN in corporate IT. Some of these technologies include the following:

- Data center bridging (DCB) Ethernet—sometimes referred to as converged enhanced Ethernet (CEE)
- Cloud technologies
- Locally attached flash storage

Cloud storage and cloud everything is squeezing its way into many IT environments, usually starting at the bottom but working its way to the top. Common examples include test and development environments moving to the cloud, often leading to staging and some production requirements eventually ending up in the cloud. At the top, business-critical line-of business applications—which are more immune to the cloud because of their high-performance, low-latency requirements—are benefitting from locally attached flash storage—where flash is placed on the server's PCIe bus, putting it close to the host CPU and memory. That just leaves the rest of the production environment that is too important to be trusted to the cloud but doesn't require ultra-low latency. But even this use case is under pressure from FCoE, which maps FCP over 10 GB DCB Ethernet. All in all, the FC SAN is under extreme pressure, and it should be! But it isn't about to go away overnight or without a fight. A good set of FC skills will serve you well in FCoE environments in the future.

Next let's take a look at some of the major technology components that make up an FC SAN.

FC SAN Components

FC SANs are made up of several key components—some physical, some logical. In this section, you'll consider the following, and more, in greater detail:

- Host bus adapters and converged network adapters
- FC switches and directors
- FC storage arrays
- FC cabling
- FC fabrics
- FC simple name server
- Zoning
- FC addressing
- FC classes of service
- VSAN technology

Physical Components

At a very high level, a SAN comprises end devices and switches. The *end devices* communicate via the *switches*.

We usually think of end devices—sometimes called *node ports,* or *N_Ports*—as servers and storage arrays. However, to be technically accurate, end devices are ports on HBAs and CNAs installed in servers, storage arrays, and tape devices. End devices connect to switches via cables. Sadly, data center networking technologies have not made the leap to wireless. FC SANs use optical fiber cables. Multiple switches can be linked together to form larger *fabrics*.

NOTE More often than not, this book refers to an FC SAN as simply a *SAN.*

Anyway, those are the basics. Let's take a closer look at each of the FC SAN components.

HBAs and CNAs

Hosts and servers connect to the SAN via one or more Fibre Channel host bus adapters (HBAs) or converged network adapters (CNAs) that are installed on the PCIe bus of the host. In SCSI parlance, these are the *initiators* on the SAN.

As mentioned earlier, HBAs and CNAs can be implemented either as PCIe expansion cards or directly on the motherboard of a server—a la *LAN on motherboard* (LOM).

Each HBA and CNA has its own dedicated hardware resources that it uses to offload all FCP-related overhead from the host CPU and memory. This offloading is a key reason that FC SAN is a fast, low-overhead, but relatively expensive, storage networking technology.

Also, most HBAs and CNAs come with a BIOS, meaning that the servers they are installed in can *boot from SAN*.

NOTE

Technically speaking, CNAs are multiprotocol adapters that provide hardware offloads for multiple protocols, including FC, iSCSI, and IP. This means they can be used for high-performance FC SAN, iSCSI SAN, and Ethernet networking. This flexibility offers an element of future-proofing for your environment; today you might want your CNA to operate as an iSCSI initiator, but in six months' time, if the server is rebuilt, you might need to configure the CNA as an FC initiator. However, as you might expect, this flexibility and future-proofing has a price tag to reflect its convenience! In the real world, however, CNAs are almost exclusively used as FCoE adapters. iSCSI offloads are rarely, if ever, implemented and used.

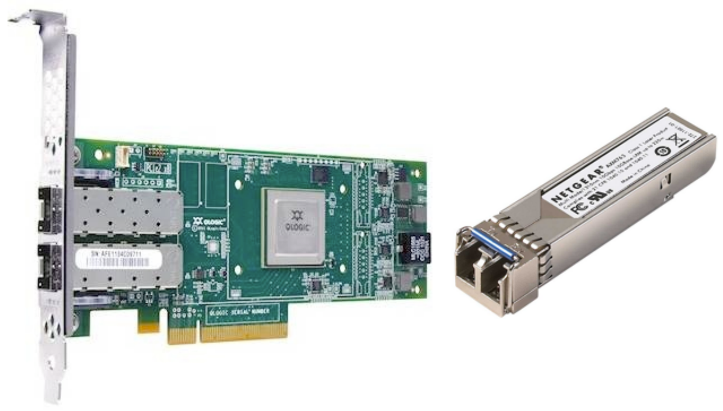
As far as hypervisors and operating systems are concerned, HBAs and CNAs show up on the PCI device tree as SCSI adapters that provide access to standard raw SCSI devices known as LUNs. The OS and hypervisor are unaware that the storage devices they are accessing are over a shared network.

NOTE

The term *LUN* is a throwback from SCSI, referring to *logical unit number*. Without getting into the depth of SCSI, in order for devices on a SCSI bus (or FCP SAN) to be addressed, they need a LUN. The term *LUN* and *volume* are often used interchangeably. So we may sometimes refer to LUNs as volumes, and vice versa.

HBAs and CNAs often support either copper direct-attach cables (DACs) or optical transceivers. When you buy the HBA or CNA, you typically have to specify which of the two you want. Generally speaking, optical costs more but supports longer distances, requires less power, and is more popular. These transceivers are often either SFP or SFP+ modules. *SFP* stands for *small form-factor pluggable*, and *SFP+* is an enhanced version of SFP that supports higher data rates, including 10 GB Ethernet. Both types of transceiver are hot-pluggable. Figure 5.2 shows an HBA with optical transceivers as well as an SFP optic.

FIGURE 5.2 HBA and SFP optics



FC Switches and Directors

FC switches and *FC directors* are essentially the same thing—physical network switches that contain multiple physical ports and that support the FC Protocol.

Switches and directors provide connectivity between end devices such as hosts and storage. They operate at layers FC-0, FC-1, and FC-2 and provide full bandwidth between communicating end devices. In addition, they provide various *fabric services* that simplify management and enable scalability. Also, if multiple FC switches are properly networked together, they merge and form a single common fabric.

The difference between directors and switches is by convention only, so don't get hung up on it! Convention states that larger switches, usually with 128 or more ports, are referred to as *directors*, whereas those with lower port counts are referred to as merely *switches*, *departmental switches*, or *workgroup switches*. To be honest, most people these days just use the term *switch*.

That said, directors have more high-availability (HA) features and more built-in redundancy than smaller workgroup-type switches. For example, director switches have two control processor cards running in active/passive mode. In the event that the active control processor fails, the standby assumes control and service is maintained. This redundant control processor model also allows for nondisruptive firmware updates. Smaller workgroup switches do not have this level of redundancy.

Directors tend to be blade-based architectures that allow you to pick and choose which blade types to populate the director chassis with. These blades come in varying port counts (such as 48-port or 64-port) as well as offering different services (such as 16 Gbps FC switching, 10 GB FCoE-capable DCB Ethernet networking, encryption capabilities, and so on).

Before you start making any changes regarding your blades, you need to determine which slots in a director switch are populated, as well as what kind of blades each slot is populated with. Exercise 5.1 shows how you can easily do this.

EXERCISE 5.1

Using hashow and slotshow in Brocade Switches

This exercise walks you through checking the population of slots and types of blades in a director switch. This exercise is specifically checking the slots in a Brocade DCX director switch. Your system and results might be different, but the general principles presented here will probably still apply.

1. Before you get started, it might be a good idea to check the HA status of your blades to see how healthy everything is. You can do this by running the hashow command:

```
LegendarySw01:admin> hashow
Local CP (Slot 6, CP0): Active, Warm Recovered
Remote CP (Slot 7, CP1): Standby, Healthy
HA enabled, Heartbeat Up, HA State synchronized
```

These results show the HA status of the two control processor (CP) blades installed in a Brocade DCX director switch. As you can see, the CP in slot 6 is currently active, while the CP in slot 7 is in standby mode. The two CPs are currently in sync, meaning that failover is possible.

2. Now run the `slotshow` command, which shows which slots in a Brocade DCX director switch are populated, as well as what kind of blades each slot is populated with:

```
LegendarySw01:admin> slotshow
Slot Blade Type    ID Model Name Status
--------------------------------------------------
1    SW BLADE      77 FC8-64     ENABLED
2    SW BLADE      97 FC16-48    ENABLED
3    SW BLADE      96 FC16-48    ENABLED
4    SW BLADE      96 FC16-48    ENABLED
5    CORE BLADE    98 CR16-8     ENABLED
6    CP BLADE      50 CP8        ENABLED
7    CP BLADE      50 CP8        ENABLED
8    CORE BLADE    98 CR16-8     ENABLED
9    UNKNOWN                     VACANT
10   UNKNOWN                     VACANT
11   UNKNOWN                     VACANT
12   UNKNOWN                     VACANT
```

As you can see, this director switch has 8 of its 12 slots populated. It has three 48-port 16 Gbps–capable switching blades, a single 64-port 8 Gbps switching blade, two core blades, and two control processor blades.

Switch Ports

FC switches and directors contain multiple ports of varying types. At a physical level, these port types will be either optical or copper SFP/SFP+ ports supporting either FCP or DCB Ethernet/FCoE. Optical is far more popular than copper, but we'll discuss this in more detail soon. As mentioned when we discussed HBAs and CNAs, SFP is an industry-standard pluggable connector.

The following `sfpshow` command on a Brocade switch shows detailed information about SFP modules installed in the switch:

```
LegendarySw01:admin> sfpshow
Slot 1/Port 0: id (sw) Vendor: HP-A BROCADE
Serial No: UAA110130000672U Speed: 2,4,8 MB/s
```

From this, you can see that this SFP is seated in port 0 in slot 1. You can also see that it has a shortwave (sw) laser, you can see its serial number, and you can see that it supports transmission speeds of 2, 4, and 8 MB/s.

Switch ports also have logical parameters. In an FC SAN, a switch port can be in any one of many modes. The most important and most common of these modes are explained here:

U_Port This is the state that FC ports are typically in when unconfigured and uninitialized.

N_Port This is the node port. End devices in a fabric such as ports in server or storage array HBAs log in to the fabric as N_Ports. Host N_Ports connect to switch F_Ports.

F_Port Switch ports that accept connections from N_Ports operate as fabric ports.

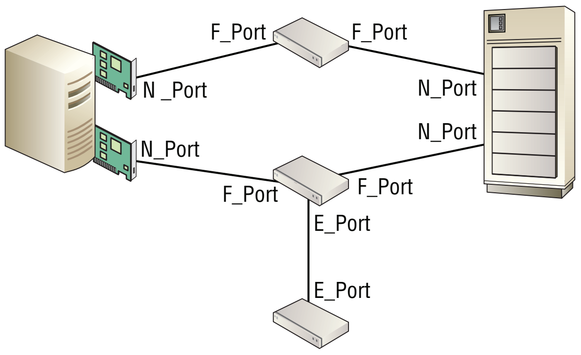
E_Port The expansion port is used to connect one FC switch to another FC switch in a fabric. The link between the two switches is called an ISL.

EX_Port This is a special version of the E_Port used for FC routing. The major difference between E_Ports and EX_Ports is that E_Ports allow the two switches to merge fabrics, whereas EX_Ports prevent fabrics from merging.

There are other port modes, but those listed are the most common.

Figure 5.3 shows N_Port, F_Port, and E_Port connectivity.

FIGURE 5.3 N_Port, F_Port, and E_Port connectivity



Port Speed

As you saw earlier in the output of the `sfpshow` command, switch ports can operate at different speeds. Common FC speeds include the following:

- 2 Gbps
- 4 Gbps
- 8 Gbps
- 16 Gbps

FCoE/DCB Ethernet ports commonly operate at the following:

- 10 GB
- 40 GB

FC ports—HBA ports, switch ports, and storage array ports—can be configured to *autonegotiate* their speed. Autonegotiate is a mini-protocol that allows two devices to agree on a common speed for the link. More often than not, this works, with the highest possible speed being negotiated. However, it does not always work, and it is good practice in the real world to *hard-code* switch-port speed at both ends of the link. For example, if a host HBA that supports a maximum port speed of 8 Gbps is connected to a switch port that supports up to 16 Gbps, you would manually hard-code both ports to operate at 8 Gbps. This practice may be different for FCoE switches and blades that have DCB ports running at 10 GB or higher. For FCoE use cases, consult your switch vendor's best practices.

Domain IDs

Every FC switch needs a domain ID. This *domain ID* is a numeric string that is used to uniquely identify the switch in the fabric. Domain IDs can be administratively set or dynamically assigned by the principal switch in a fabric during a reconfigure fabric event. The important thing is that a domain ID must be unique within the fabric. Bad things can happen if you configure two switches in the same fabric with the same domain ID.

WARNING

If you use port zoning, which we cover shortly, changing a switch's domain ID will change all the port IDs for that switch, requiring you to update your zoning configuration. That is a nightmare.

Principal Switches

Each fabric has one, and only one, *principal switch*. The principal switch is the daddy when it comes to the pecking order within a fabric. What the principal switch says, goes!

Principal switches are responsible for several things in a fabric, but the things that will be most pertinent to your day job are the following:

- They manage the distribution of domain IDs within the fabric.
- They are the authoritative source of time in a fabric.

Aside from these factors, you shouldn't have to concern yourself with the role of the principal switch in your day job.

Native Mode and Interoperability Mode

FC fabrics can operate in either native mode or interoperability mode. Your switches will be operating in native mode 99.9 percent of the time, and this is absolutely the mode you want them operating in.

Interop modes were designed to allow for heterogeneous fabrics—fabrics containing switches from multiple vendors. In order to allow for these heterogeneous configurations, interop modes disable some advanced and vendor-specific features. Use cases for interop mode tend to be restricted to situations such as acquisitions, where two companies' IT environments come together and have fabrics running switches from different vendors. But even in those circumstances, merging such fabrics and running in interop mode is rarely a good experience. The best advice to give on interop modes is to avoid them at all costs!

However, if you do need to operate fabrics with switches from different vendors, there is now a much better way than using interop modes. A new technology known as NPV has come to the rescue, and we'll talk about it later in the chapter.

So, with what you have learned so far, the following `switchshow` command on `LegendarySw01` tells us a lot of useful information:

```
LegendarySw01:admin> switchshow
SwitchNAme:   LegendarySw01
SwitchType:   62.3
SwitchState:  Online
SwitchMode:   Native
SwitchRole:   Principal
SwitchDomain: 44
SwitchId:     fffXXX
SwitchWwn:    10:00:00:05:AA:BB:CC:DD
Zoning:       ON (Prd_Even1)
SwitchBeacon: OFF

Index Port Address Media Speed State     Proto
==============================================
  0   0    220000  cu    8G    Online    FC  F_Port 50:01:43:80:be:b4:08:62
  1   1    220100  cu    8G    Online    FC  F_Port 50:01:43:80:05:6c:22:ae
```

FC Hubs

As with the rest of the networking world, FC hubs have been superseded by switches, and that is good. Switches are far more scalable and performant than hubs.

Technically speaking, FC hubs operated at the FC-0 layer and were used to connect FC-AL (arbitrated loop) devices. Due to FC-AL addressing limitations, hubs could address only 127 devices, but in practice were even smaller. FC-AL hubs also required all devices to share bandwidth and arbitrate for control of the wire, ensuring that only a single device could communicate on the hub at any one point in time.

Switches, on the other hand, operate up to the FC-2 layer, provide massive scalability, allow multiple simultaneous communications, and provide full nonblocking bandwidth to all connected devices.

Thankfully, FC hubs are generally a thing of the past, and you are unlikely to see them anywhere other than a computer history museum.

FC Storage Arrays

As this book contains an entire chapter dedicated to storage arrays, this section touches on them only lightly.

In an FC SAN, a *storage array* is an end point/end device with one or more node ports (N_Ports). These node ports are configured in Target mode—and therefore act as SCSI targets—accepting SCSI commands from SCSI initiators such as server-based HBAs and CNAs. Because they are node ports, they participate fully in FC SAN services, such as registering with and querying the name server and abiding by zoning rules.

Storage arrays tend to house lots (hundreds or even thousands) of disk and flash drives that are shared over the FC SAN as block devices referred to as either *LUNs* or *volumes*.

NOTE For a deep dive into storage arrays, see Chapter 3, "Storage Arrays."

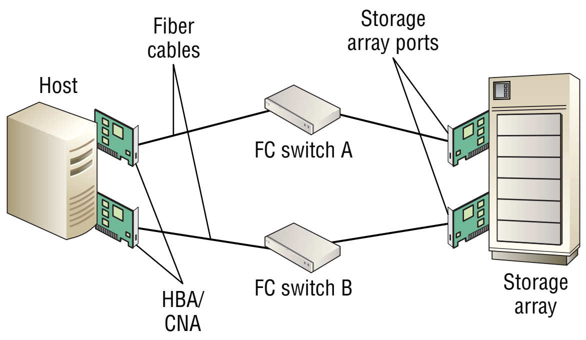
Cables

Don't overlook cabling, as it is a critical component of any SAN. Sadly, FC hasn't made the jump to wireless. Nor have any other major data-center networking technologies, for that matter. Therefore, cables and cabling are pretty damn important in SAN environments. Let's kick off with a few basics and then get deeper.

Almost all FC SAN cabling is optical fiber. Optical fiber transmits data in the form of light that is generated by lasers or LEDs. Copper cabling can be used, but this is rare compared to optical. Optical may be used in Top of Rack (ToR) or End of Row (EoR) situations. Some storage vendors used copper cabling on their old FC backends before moving to SAS backends. All optical fiber cabling consists of two pairs of fibers—transmit (TX) and receive (RX)—allowing for full-duplex mode.

Figure 5.4 shows a simple FC SAN with two end devices and an FC switch, and points out the location of some physical components.

FIGURE 5.4 Simple FC SAN physical components



Right then, let's get into some detail. Optical fiber cables come in two major flavors:

- Multi-mode fiber (MMF)
- Single-mode fiber (SMF)

Multi-mode fiber transmits light from shortwave lasers, whereas single-mode fiber transmits light from a longwave laser.

When it comes to SAN cabling, multi-mode fiber is the undisputed king of the data center. This is thanks to two things:

- Multi-mode fiber is cheaper than single-mode.
- Multi-mode fiber can easily transmit data over the kinds of distances common to even the largest of data centers.

Single-mode fibers and their associated longwave lasers (optics) are more expensive and tend to be used by telecommunications companies that require longer cable runs, such as hundreds of miles.

All fiber cables—multi-mode and single-mode— are made up of the following major components:

- Core
- Cladding
- Jacket

The *core* is the insanely small glass or plastic cylinder at the center of the cable that carries the light. If you have seen a stripped fiber cable and think you have seen the core of the cable with your naked eye, you are either Superman or you saw something that wasn't the core. The core of a fiber cable is far too small to be seen by the naked eye!

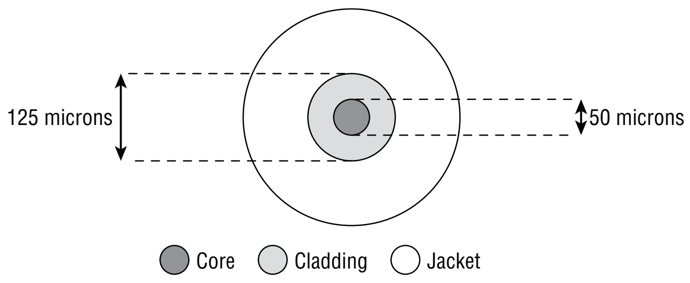
Surrounding the core is the *cladding*. The cladding protects the core and ensures that light does not escape from the core.

The outer sheath of all fiber cables is referred to as the *jacket*. Jackets are usually color coded according to the type of fiber they carry.

Yes, there are other layers and substances between the cladding and the jacket. However, knowledge of them is of no value to you in your day-to-day job.

Figure 5.5 shows the core, cladding, and jacket of a fiber-optic cable.

FIGURE 5.5 Core, cladding, and jacket

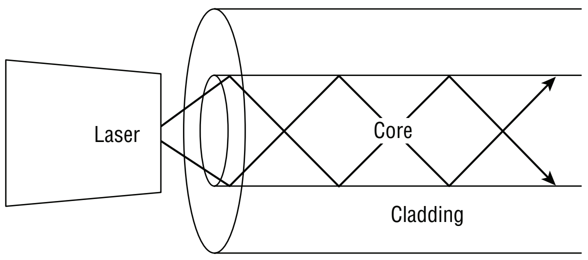


Multi-Mode Fiber

Multi-mode fiber has a relatively large core diameter compared to the wavelength of the light that it carries, allowing the electronics and lasers that generate the light to be less precise, and therefore cheaper, than those used with single-mode fiber.

As light is injected onto a multi-mode fiber, it can enter the cable at slightly different angles, resulting in the light waves bouncing off the core/cladding perimeter as it travels the length of the cable. This is shown in Figure 5.6.

FIGURE 5.6 Multi-mode fiber carrying two beams of light



However, over distance, these discrete light beams crisscross, resulting in some signal loss referred to as *attenuation*. The longer the cable is, the more crisscrossing of light beams occurs, resulting in increased attenuation. Multi-mode fiber can't do very long distances.

Multi-mode fiber comes in the following two core/cladding options:

- 62.5/125 μm
- 50/125 μm

More often than not, these are referred to as *62.5 micron* and *50 micron*, respectively. 62.5 and 50 refer to the cross-sectional size of the core in microns. A 50 µm core allows light to be sent more efficiently than 62.5, so it is now more popular than 62.5.

The relatively large core size of multi-mode cables—obviously incredibly small, but large compared to single-mode—means that they can be manufactured to a slightly lower-quality, less-precise design than single-mode cables. This makes them cheaper than single-mode.

Multi-mode fibers are also categorized by an optical multi-mode (OM) designator:

- OM1 (62.5/125)
- OM2 (50/125)
- OM3 (50/125) sometimes referred to as *laser optimized*
- OM4 (50/125)

By convention, the OM1 and OM2 jacket color should be orange, whereas the OM3 and OM4 jacket color should be aqua (light blue). However, most fiber cables can be purchased in just about any color you wish, and some people choose color schemes that make managing their data center easier. One example is having one cable color for one fabric and another cable color for the other fabric. This will stop you from mixing up your cabling between fabrics. However, a better option might be color-coding the cable connectors rather than the actual cable jackets.

Typically, each time the speed of FCP is increased, such as from 4 Gbps to 8 Gbps, or from 8 Gbps to 16 Gbps, the maximum transmission distance for a given cable type drops off. Table 5.1 shows the maximum distances supported by different multi-mode cable types at the various common FCP speeds.

TABLE 5.1 MMF distances

FC Speed (Gbps)	OM1	OM2	OM3	OM4
4 Gbps	70 m	150 m	380 m	400 m
8 Gbps	21 m	50 m	150 m	190 m
16 Gbps	15 m	35 m	100 m	125 m

Table 5.1 should help you know which cable type you need to buy. While some companies are now opting for OM3 everywhere, doing this will see you paying more money unnecessarily if you are running cables within a rack or between adjacent racks. On the other hand, using OM3 everywhere for structured cable runs might be a good idea as this future-proofs your infrastructure for things like 40 GB and 100 GB Ethernet. However, intra-cabinet and adjacent cabinet runs do not yet require OM3, and if at some point in the future they do, they can be easily ripped and replaced.

Single-Mode Fiber

Single-mode fiber cables carry only a single beam of light, typically generated by a long-wave laser. Single-mode fiber cables are narrow, relative to the wavelength of the laser, meaning that only one mode is captured and therefore there isn't the interference or the excessive reflecting off the core/cladding boundary while light travels the length of the cable. This results in single-mode fiber being able to carry light over distances measured in miles rather than feet. Single-mode fiber cables also tend to be more expensive than multi-mode.

Single-mode fiber core size is usually 9 μm but still with a 125 μm cladding. This 9 μm core requires extremely precise engineering and is a major reason for single-mode fiber being more expensive than multi-mode.

Single-mode fiber is less popular in data centers than multi-mode, and commonly has a yellow jacket.

Bend Radius and Dust

Because fiber cables have a glass core, you need to be really careful when bending them. Bend them too far, and you will break them. And they're not cheap, so you've been warned!

When talking about bending fiber cables, we use the term *bend radius*. The *minimum bend radius* is as tight as a cable can be bent without seriously damaging it.

If you bend the cable too tightly, you can often leave a visible kink in the cable, making it easy to see a broken cable. However, even if you don't kink the cable or crack the glass core, you can stress the cladding so much that it loses its refractive capability—its ability to keep light within the core. Either way, the cable is damaged and will only give you a headache if you continue to use it.

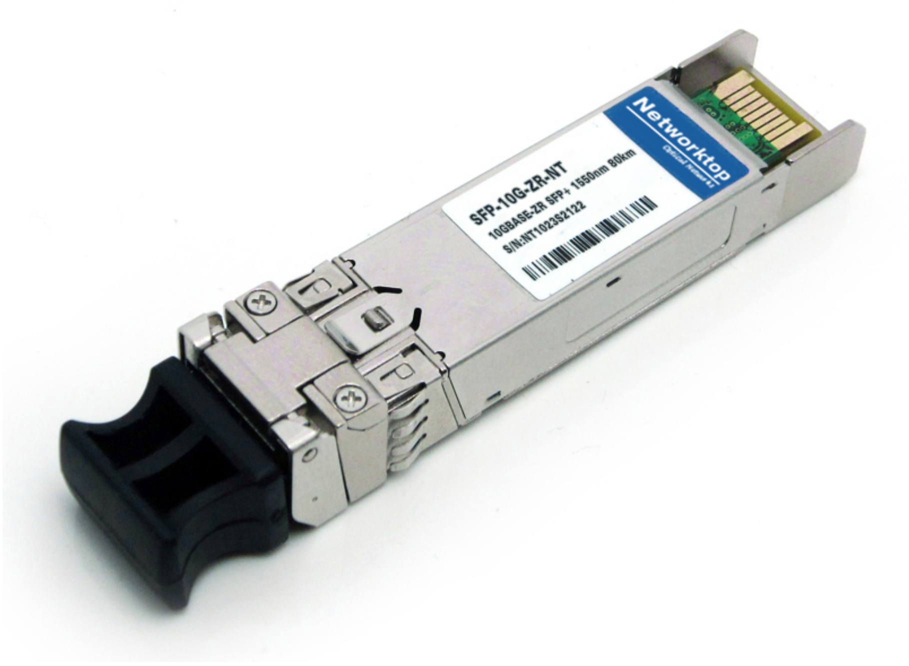
The recommended minimum bend radius for a fiber cable varies depending on the type of fiber cable—OM1 differs from OM3, single-mode differs from multi-mode, and so on. As a general rule, the larger the bend radius the better, a general rule of no less than 3 inches would be a good place to start. That way, you reduce the risk of damage from bending. Make sure you *always* adhere to the manufacturer's recommendation and don't take risks!

We should also point out that dust on the end of a cable connector or on an optical switch port can cause issues such as flaky connections or connections that log high error counts. This is why new fiber cables come with a rubber or plastic cap—to keep dust out. Not enough people follow the good practice of properly cleaning the tips of fiber cables before using them. Keep your cable ends clean! Figure 5.7 shows an SFP optic with a rubber dust cover.

WARNING

Cleaning fiber cable connectors has to be done properly with the appropriate equipment. Do not spit on the end of a fiber-optic cable and rub it clean with the end of your sleeve! There are several options available on the market.

FIGURE 5.7 SFP+ module with rubber dust cap



Cable Connectors

Most fiber cables come with male connectors that allow them to be easily popped into and out of switch, HBA, and storage array ports—a process sometimes called *mating*. While lots of fiber connector types are available, there are a few that you will need to be familiar with as storage administrator, so we'll take a look at these.

SC and LC are probably the most common connectors. *SC* stands for *standard connector*, and *LC* stands for *lucent connector*. Both are male connectors. LC connectors are only half the size of SC connectors, and for this reason are becoming more and more popular. Half the size can equal double the density.

NOTE

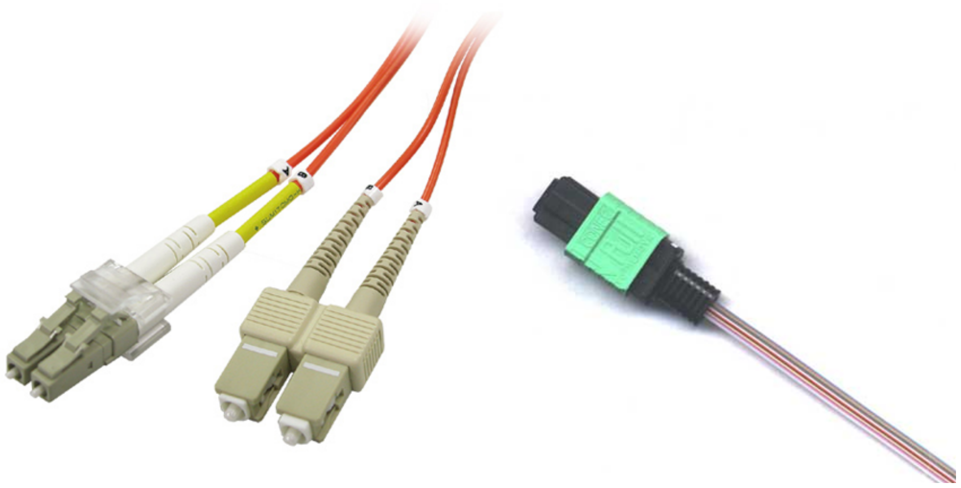
LC connectors are half the size of SC connectors, and both are available in multi-mode and single-mode cables.

Another commonly used cable is the MPO or MTP cable. *MTP*, which stands for multi-fiber termination push-on, is a ribbon cable that carries multiple fibers (usually 12 or 24) and terminates them at a single connector. MTP is a variation of the MPO connector.

Other connector types do exist, but SC, LC, and MTP are by far the most popular in most modern data centers.

Figure 5.8 shows SC, LC, and MTP cable connectors.

FIGURE 5.8 Common fiber cable connectors



If fiber cables do not come with a connector, the only way to connect two together is to *splice* them. Splicing is a specialized procedure requiring specialized equipment, usually performed by specialist personnel. You may well go your entire storage career without having to splice cables.

It is time to move on to the logical components of FC SANs.

Logical SAN Components

Now let's talk about some of the theory and logical components of the SAN.

FCP Stack

Like all good networking technologies, FCP is a layered protocol, only it's not as rich as protocols like Ethernet and TCP/IP. The net result is a simpler stack. Figure 5.9 shows the layers of the Fibre Channel Protocol.

FIGURE 5.9 FCP stack

FC-4	Upper-layer protocols (PDU): SCSI, IP
FC-3	Not implemented
FC-2	Framing, flow control, zoning, ordered sets
FC-1	Signalling, encode/decode including 8b/10b
FC-0	Lasers, cables, connectors, data rates (x Gbps)

Because FCP is a simpler protocol, you will find yourself referring back to the FCP layers less often than you do for a TCP/IP network.

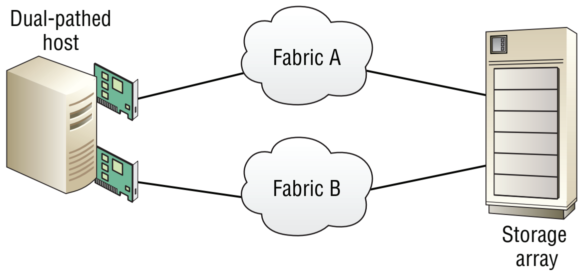
Just in case the IP network team try to you tell you that FCP's simplicity or fewer layers is some sort of shortcoming for FCP, it is not. It is actually a reflection of the fact that FCP is more of a *channel* technology than a *network* technology. Remember at the beginning of the chapter we explained that *channels* are low-overhead, low-latency, high-speed interconnects that attempt to re-create channels rather than massively complex networks.

FC Fabrics

A *fabric* is a collection of connected FC switches that have a common set of services. For example, they share a common name server, common zoning database, common FSPS routing table, and so on.

Most of the time you will want to deploy dual redundant fabrics for resiliency. Each of these two fabrics is independent of the other, as shown in Figure 5.10.

FIGURE 5.10 Dual redundant fabrics



NOTE: It is common practice to draw a fabric as a cloud.

Each fabric is viewed and managed as a single logical entity. If you want to update the zoning configuration in the fabric, it is common across the fabric and can be updated from any switch in the fabric.

If you have two switches that are not connected to each other, each switch will run its own isolated fabric. If you then connect these two switches by stringing a cable between an E_Port on each switch, the fabrics will *merge* and form a single fabric. How this merging happens, and what happens to existing zoning configurations and principal switches, can be complicated and depends on the configuration of your fabrics. Before doing something like this, do your research and follow your vendor's best practices.

It is possible that two fabrics won't merge. When this happens, the fabrics are said to be *segmented*. Switches segmenting from fabrics can happen if certain important configuration parameters in each switch's configuration file do not match. Before merging fabrics, make sure that all switches in the fabric are configured appropriately.

Once the fabrics have merged, any device in the fabric can communicate with any other device in the fabric, as long as the zoning configuration allows it.

NOTE: Switches merge when connected by two E_Ports—one on each switch. The link formed between these two E_Ports is referred to as an inter-switch link, or ISL for short. However, many switch ports dynamically configure themselves as E_Ports if they detect another switch on the other end of the cable. You need to be aware of this, as you may unintentionally merge two fabrics if you get your cabling wrong. We talk about ISLs in detail when we discuss fabric topologies.

Fabric Services

All switches in a fabric support and participate in providing a common set of fabric services. As defined by FC standards, these fabric services can be reached at the following well-known FC addresses:

0xFF FF F5: Multicast server

0xFF FF F6: Clock sync server

0xFF FF FA: Management server

0xFF FF FB: Time server

0xFF FF FC: Directory server / name server

0xFF FF FD: Fabric controller server

0xFF FF FE: Fabric login server

0xFF FF FF: Broadcast address

Not all of these fabric services are implemented, and from a day-to-day perspective, some are more important than others. Let's take a closer look.

The *time server* is responsible for time synchronization in the fabric. The *management server* allows the fabrics to be managed from any switch in the fabric. The zoning service is also a part of the management server. The *fabric controller server* is responsible for principal switch selection, issuing of registered state change notifications (RSCNs), and maintaining the Fabric Shortest Path First (FSPF) routing table. The *fabric login server* is responsible for issuing N_Port IDs and maintaining the list of registered devices on the fabric.

FC Name Server

The FC *simple name server (SNS)*, better known these days as simply the *name server,* is a distributed database of all devices registered in the fabric. It is a crucial fabric service that you will interact with on a regular basis as part of zoning and troubleshooting.

All devices that join the fabric are required to register with the name server, making it a dynamic, centralized point of information relative to the fabric. As it is distributed, every switch in the fabric contains a local copy.

Targets and initiators register with, and query, the name server at the well-known address 0xFFFFFC.

Zoning

Device visibility in a SAN fabric is controlled via *zoning*. If you want two devices to be able to talk to each other, make sure they are zoned together!

Back to the original days of SCSI, when all devices were directly connected to a single cable inside a single server, that was a form of zoning—only that single server had access to the devices. Fast-forward to the days of shared SAN environments, where hundreds of targets and hundreds of initiators can talk to each other, and it's easy to see how this differs vastly from the original intent of SCSI. Something had to be done!

Fabric and Port Login

Before we dive into zoning, we'll cover some important background work on device discovery and the concept of fabric logins and port logins.

When an initiator first initializes on a SCSI bus, including an FC SAN, it performs a reset and then a discovery of all devices on the bus. In the case of a SAN, the bus is the SAN. On a SAN fabric with no zoning in place, the initiator will probe and discover all devices on the SAN fabric. As part of this discovery, every device will also be queried to discover its properties and capabilities. On a large SAN, this could take forever and be a massive waste of resources. But more important, it can be dangerous to access devices that other hosts are already using.

Real World Scenario

The Importance of Zoning

In the early days of adopting FC SANs, many companies deployed an open SAN (no zoning) or large zones with multiple initiators and targets. While this was simple, it caused a lot of problems. One such problem at a company I worked for was mysteriously resetting tape drives. The tape drives were part of a small SCSI tape library that was attached to the FC SAN. However, no zoning was in place, meaning that all devices on the small SAN could see each other and affect each other. After much head scratching and troubleshooting, the root cause of the issue was identified as SCSI resets that occurred each time a server was rebooted. Because the SAN was wide open—no zoning—any time a SCSI reset was issued, it was received by all devices on the SAN. As it turned out, the tape drives were designed to perform a reset and full media rewind on receipt of a SCSI reset. The net result was that any time a server rebooted, the ensuing SCSI reset caused all tape drives on the SAN to reset and rewind, obviously interrupting any in-progress backup jobs. This was resolved by implementing zoning.

In order to speed up, smooth out, and generally improve the discovery process, the name server was created. Since the invention of the name server, each time any device joins the fabric, it performs a process referred to as a *fabric login (FLOGI)*. This FLOGI process performs a bunch of important functions:

- Assigning devices their all-important 24-bit N_Port IDs
- Specifying the class of service to be used
- Establishing a device's initial credit stock

After a successful FLOGI process, the device then performs a *port login (PLOGI)* to the name server to register its capabilities. All devices joining a fabric must perform a PLOGI to the name server. As part of the PLOGI, the device will ask the name server for a list of devices on the fabric, and this is where zoning comes into play. Instead of returning a list of all devices on the fabric, the name server returns only a list of those devices that are zoned so as to be accessible from the device performing the PLOGI. Once the device performing the login has this list, it performs a PLOGI to each device in order to query each device's capabilities. This process is much quicker and more secure than probing the entire SAN for all devices, and it also allows for greater control and more-flexible SAN management.

Even with the preceding process, if no zoning is in place on the fabric, it is literally a free-for-all. Every device can see every other device, and you will find yourself in a very bad place.

Now let's take a closer look at zoning.

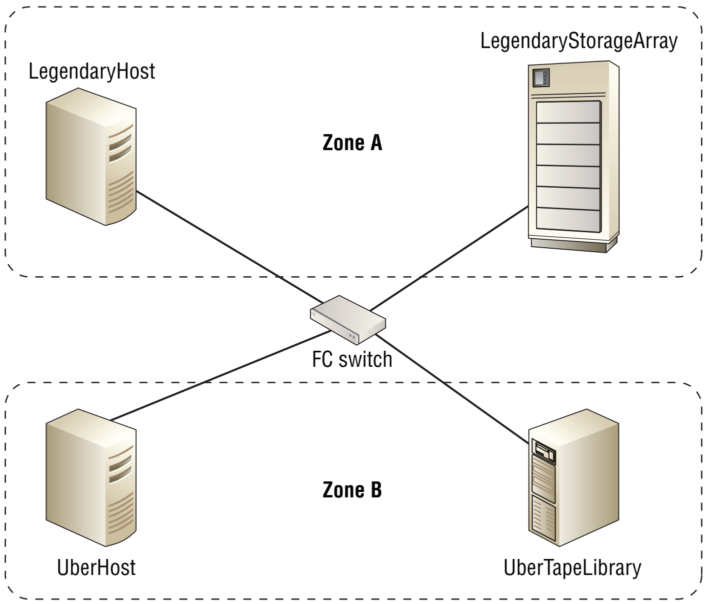
Zones

The basic element of zoning is the *zone*¸ and all SAN fabrics will contain multiple zones. Multiple zones are grouped together into a container know as a *zone set*.

As we've hinted already, zoning is a way of partitioning a fabric, with each individual zone being like a mini fabric, a subset of the wider fabric. If you want two devices to communicate with each other, put them in the same zone; all other devices should be excluded from that particular zone.

Figure 5.11 shows a simple SAN with two zones (Zone A and Zone B). In the figure, `LegendaryHost` can communicate with only `LegendaryStorageArray`, and at the same time, `UberHost` can communicate with only `UberTapeLibrary`. There are four devices on the fabric and two zones in operation—kind of like two mini fabrics.

FIGURE 5.11 Simple zoning example



Although the example in Figure 5.11 is small, zoning scales to extremely large fabrics with potentially thousands of devices and thousands of zones. As in this example, zones are made up of initiators and targets. It is standard practice to give these initiators and targets friendly names, referred to as *aliases*, which can help immensely when troubleshooting connectivity issues.

Exercise 5.2 shows how to configure zoning.

EXERCISE 5.2

Configuring Zoning

For simplicity, you will create a single zone in a single fabric. More often than not, you will deploy dual redundant fabrics and create a zone in each fabric so that each server can access its storage via two discrete fabrics.

1. Create an alias for a new host you have added to your SAN. The new host is called `LegendaryHost`, and its HBA has a WWPN of `50:01:43:80:06:45:d3:2f`. In this example, you will call the alias `LegendaryHost_HBA1`, as it is an alias for the WWPN of the first HBA in our server called `LegendaryHost`.

   ```
   alicreate LegedaryHost_HBA1 “50:01:43:80:06:45:d3:2f”
   ```

   Assume that the alias for your EMC VMAX storage array port is already created in the fabric and is called `VMAX_8E0`.

2. You will now create a new zone containing two aliases. One alias will be the new alias you just created, and the other will be for port 8E0 on your EMC VMAX storage array. The new zone will be called `LegendaryHost_HBA1_VMAX_8E0` so that you know the purpose of the zone when you look at its name. Use the `zonecreate` command to create the zone:

   ```
   zonecreate LegendaryHost_HBA1_VMAX_8E0, "LegedaryHost_HBA1; VMAX_8E0"
   ```

   You now have a new zone containing two aliases that will allow `HBA1` in `LegendaryHost` to communicate with port 8E0 on your EMC VMAX array.

3. Add the newly created zone to your existing zone set called `Prod_Config` by using the `cfgadd` command:

   ```
   cfgadd Prod_Config, LegendaryHost_HBA1_VMAX_8E0
   ```

4. Save the configuration.

   ```
   cfgsave
   ```

5. Make sure that `Prod_Config` is the active zone set:

   ```
   cfgactvshow
       Effective Configuration:
       cfg: Prod_Config
       <output truncated>
   ```

This procedure created a new alias for `HBA1` in `LegendaryHost` and created a new zone containing the aliases for the WWPNs of `LegendaryHost` and our EMC VMAX array port 8E0. It also added the newly created zone to the active zone set and enabled the configuration. As a result, `LegendaryHost` will now be able to communicate with our VMAX array.

Zoning Best Practices

The following rules of thumb regarding zoning will keep your SAN in good shape:

1. Keep zones small. This will make troubleshooting simpler.
2. Have only a single initiator in each zone. It is considered bad practice to have more than one initiator in a zone.
3. Keep the number of targets in a zone small, too. It is more acceptable to have multiple targets per zone than it is to have multiple initiators per zone. However, don't get carried away, as this can make troubleshooting more difficult.
4. Give your zones and aliases meaningful names.
5. Take great care when making zoning changes in your environments.

Of these best practices, the one that is probably the most important in the real world is point 2, referred to as *single-initiator zoning*. In a nutshell, single-initiator zoning states that every zone should contain only a single initiator plus any targets that the initiator needs to access. Live by this principle, and your life will be made a lot easier.

NOTE

It's important to understand that initiators and targets can be members of multiple zones. In fact, this is necessary if you want to implement the best practice of single-initiator zones, but it also allows multiple initiators to access a single target. For example, if you have two initiators wanting to access the same target, you would create two zones: one zone containing the first initiator and the target, and the second zone containing the second initiator and the same target. This way, you have two separate zones, each with only a single initiator, but both zones have the same target. The end result of this approach is that two initiators are able to access a single target, but it also maintains the best practice of implementing single-initiator zones.

Zone Sets

Multiple zones are grouped together in to *zone sets*, and it is the zone set that is applied to the fabric. If you have configured a new zone, you will need to add it to the active zone set in order for it to be applied to the fabric.

Real World Scenario

Don’t Forget to Apply Your Zoning Changes!

A SAN administrator was confused when new zones didn’t appear to be working. After a lot of double-checking and research, the administrator eventually realized that he had been creating new zones and saving them to the active zone set and leaving it at that. In order for newly created zones to be applied to the fabric, they need to be added to the active zone set, and the updated active zone set usually needs reapplying to the fabric. Forgetting to reapply the active zone set to the fabric is a common mistake in many organizations.

It is possible for an FC fabric to have multiple defined zone sets. However, only a single zone-set can be active on any given fabric at any given time.

Aliases

We’ve briefly mentioned aliases, but let’s have a closer inspection.

WWPNs are hideously long and mean nothing to the average human being. Aliases allow us to give friendly names to WWPNs on the SAN. Instead of having to remember that WWPN `50:01:43:80:06:45:d3:2f` refers to `HBA1` in our host called `LegendaryHost`, we can create an alias for that WWPN with a meaningful name.

The following command creates an alias called `LegendaryHost_HBA1` in a Brocade fabric:

```
LegendarySw01:admin> alicreate LegedaryHost_HBA1 "50:01:43:80:06:45:d3:2f"
```

The following set of commands creates the same alias, but this time on a Cisco MDS fabric:

```
MDS-sw01# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
MDS-sw01 (config)# fcalias name LegendaryHost_HBA1
MDS-sw01 (config-fcalias)# member pwwn 50:01:43:80:06:45:d3:2f
MDS-sw01 (config-fcalias)# exit
```

Most people use aliases when configuring zoning, and it is a solid practice to stick to.

Port or WWN Zoning

There are two popular forms of zoning:

- WWN zoning
- Port zoning

Essentially, these are just two ways of identifying the devices you are zoning—either by WWPN or by the ID of the switch port the device is connected to.

Each approach has its pros and cons. If you go with port zoning and then have to replace a failed HBA in a host, you will not need to update the zoning as long as the new HBA remains cabled to the same switch port that the failed HBA was. However, if you implement WWN zoning, you will need to either update the alias and reapply the zone set to reflect the new WWPN of the new HBA or reset the WWPN on the new HBA to match that of the old HBA. In this respect, port zoning seems simpler. However, and this is massively important, the port ID used in port zoning can change! It is rare, but it can happen. This is because port IDs contain the domain ID of the switch, and the domain ID of a switch can change if the fabric ever has to perform a disruptive reconfiguration. If this ever does happen, you are staring down the barrel of invalid zoning and a lot of hard work to get things up and running again. For this reason, most people choose to go with WWN zoning. Also, from a security perspective, port zoning is considered more secure than WWN zoning due to the act that it isn't too hard to spoof a WWN, whereas it's more difficult to spoof which port you are connected to.

NOTE Some people refer to port zoning as *hard zoning*, and WWN zoning as *soft zoning*. This can be misleading.

Hard Zoning or Soft Zoning

Soft zoning is basically name-server-enforced zoning.

When a device performs a PLOGI to the name server and requests a list of devices on the fabric, the name server returns only a list of devices that are in the same zones as the device performing the PLOGI. This way, the device logging in to the fabric gets a limited view of the fabric and *should* log in to only those devices. However, a bad driver in an HBA or CNA could decide to PLOGI to any or every possible device address on the fabric. If it did this, nothing would stop it. Therefore, soft zoning is nothing more than security by obscurity and relies on attached devices playing ball.

Both WWN and port zoning can be implemented as soft zoning.

In *hard zoning*, on the other hand, the switch hardware inspects all traffic crossing the fabric and actively filters and drops frames that are disallowed according to the zoning configuration. Name server zoning is still in effect—the name server still gives devices that are logging in a restricted view of the fabric—but hard zoning is also implemented as an additional level of security. The following command shows that on `UberSw01`, hard zoning is in place and we are using WWN zoning:

```
UberSw01:admin> portzoneshow
PORT: 0 (0)   F-Port    Enforcement: HARD WWN   defaultHard: 0  IFID: 0x44120b38
PORT: 1 (1)   Offline
PORT: 2 (2)   F-Port    Enforcement: HARD WWN   defaultHard: 0  IFID: 0x4412004b
```

Whether hard zoning is in place on your fabric is vendor specific and depends on how you have your fabric and zoning configured. Consult your vendor's best practices in order to deploy hard zoning.

NOTE

Once zoning is enforced on any fabric, any devices that do not exist in zones in the active zone set are frozen out and cannot access any devices on the fabric.

SAN Topologies

Now let's cover SAN topologies and some of the principles, such as redundancy and high availability, that underpin them.

Redundancy

A hallmark of all good storage designs is redundancy.

Look at any storage design worth the paper it was written on, and you will see redundancy everywhere: servers with multiple HBAs, cabled to diverse redundant fabrics, connected to redundant storage processors; RAID-protected disks; replicated volumes; and the list goes on and on. This is essential because although there aren't usually that many bad days in the storage world, when those bad days do come along, they tend to be days from hell!

Almost every SAN environment in the world will have dual redundant fabrics—at least two fully isolated fabrics. These are isolated physically as well as logically, with no cables between them.

TIP

When configuring redundant fabrics, make sure that each discrete fabric is on separate power circuits or at least that each switch in a fabric is powered by discrete power supplies. Also ensure that fiber cabling takes diverse routes so that dropping a floor tile on a trunk of cabling doesn't take out connectivity to both fabrics.

Figure 5.12 shows two redundant fabrics and identifies diverse cabling routes, power, and similar concerns.

Common SAN Topologies

There are several commonly recognized SAN topologies. These include point-to-point (FC-P2P), arbitrated loop (FC-AL), and the various switched fabric (FC-SW) topologies. We'll explore each of these next.

Point-to-Point

Point-to-point, referred to technically as *FC-P2P*, is a direct connection from a host HBA or CNA port to a storage array port. This connection can be a direct fly-lead connection or via a patch panel, but it cannot be via an FC switch.

FC-P2P is simple and offers zero scalability. If your storage array has eight front-end ports, you can have a maximum of eight directly attached servers talking to that storage array.

Figure 5.13 shows an FC-P2P connection.

FIGURE 5.12 Redundant cabling and power

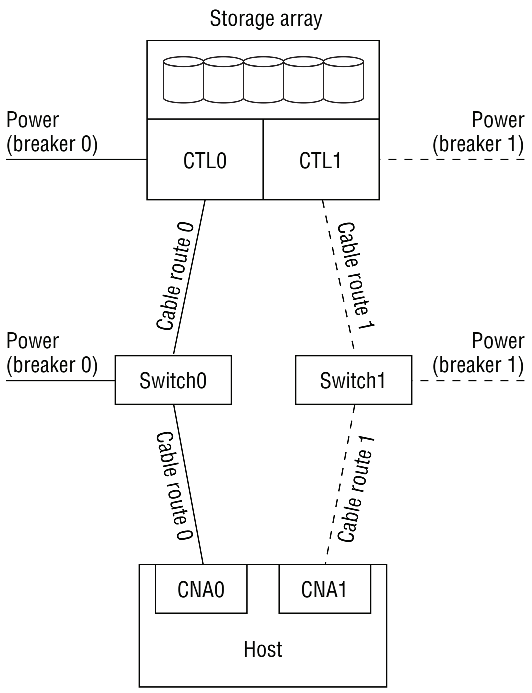
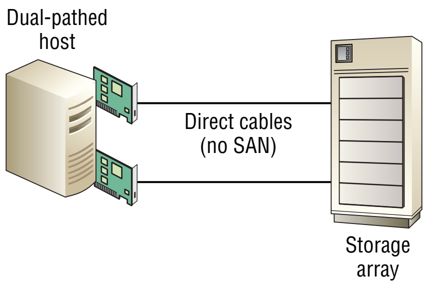


FIGURE 5.13 Simple FC-P2P configuration



Point-to-point configurations are most common in small environments where FC switches are not necessary and would unnecessarily increase costs.

Arbitrated Loop

Fibre Channel arbitrated loop, referred to as *FC-AL*, allows devices to be connected in a loop topology, usually via an FC hub. However, a hub is not required, as you can daisy-chain servers together in a loop configuration where the transmit port of one server is connected to the receive ports of the adjacent server.

FC-AL is a bit of a dinosaur technology these days and is rarely seen. This is good, because it lacks scalability (with a maximum of 127 devices) and performance.

In FC-AL configurations, all devices on the loop contend for use of the loop, and only a single device is allowed to transmit I/O on the loop at any point in time. This naturally leads to contention and drops in performance during periods of high activity.

FC-AL configurations also suffer from a phenomenon known as a LIP storm. Basically, any time a device is added to or removed from a loop, the loop reinitializes by sending loop initialization primitives (LIPs). During such reinitialization events, devices cannot transmit data on the loop. When FC-AL configurations were more commonly used, the bane of a storage administrator's life was having a flaky device on the loop that kept flooding the loop with LIP frames.

These days, FC-AL configurations mainly exist only in books like this or museums. R.I.P.

Switched Fabric

Fibre Channel switched fabric is technically referred to as *FC-SW*, but most people just call it *fabric*. It is the de facto standard in FC networking technology.

FC-SW overcomes all the limitations of FC-AL and FC-P2P, and adds a lot of improvements. As a result, this section presents it in more detail than FC-P2P and FC-AL.

A Fibre Channel fabric is made up of one or more FC switches. FC switches operate up to FC-2 layer, and each switch supports and assists in providing a rich set of fabric services such as the FC name server, the zoning database, time synchronization service, and more.

End devices can be added and removed from a fabric nondisruptively. Full nonblocking access to the full bandwidth of the fabric is provided to all devices. The fabric can scale to thousands of devices. The standards state millions of devices, but in practice this is thousands. Many FC switches operate in cut-through switching mode, which ensures the lowest possible latency by not having to buffer each frame in the switch as it passes through the switch.

When a fabric contains more than one switch, these switches are connected via a link known as an inter-switch link.

Inter-Switch Links

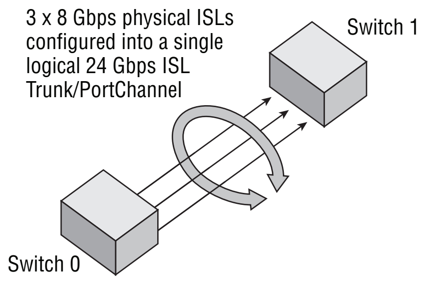
In order to connect two FC switches and form a common fabric—with a single instance of the name server and zoning table—you configure a port on each switch to be an E_Port and string a cable between these two ports. E_Ports are special ports used to connect two switches into a common fabric, and the link they form is called an *inter-switch link*, or *ISL* for short.

It is possible to run more than one ISL between two switches, and this is a common practice for redundancy. ISLs can operate as independent ISLs or can be bonded together to work as a single logical ISL. Generally speaking, bonding multiple physical ISLs into a single logical ISL is the preferred method, as this provides superior redundancy as well as superior load balancing across all physical links in the logical ISL.

NOTE Different FC switch vendors refer to the bonding of multiple physical ISLs to a single logical ISL by different names. Cisco calls it a *PortChannel*. Brocade and QLogic call it an *ISL trunk*. Referring to logical ISLs as a *trunk* can be confusing if you're not careful, as the rest of the networking world already uses the term *trunk* to refer to VLAN trunking.

Figure 5.14 shows three physical links bonded into a single logical ISL.

FIGURE 5.14 Logical ISL (PortChannel/trunk)


3 x 8 Gbps physical ISLs configured into a single logical 24 Gbps ISL Trunk/PortChannel
Switch 1
Switch 0

One important thing to consider when creating ISL PortChannels is deskew. *Deskew* refers to the difference in time it takes for a signal to travel over two or more cables. For example, it may take around 300 ns longer for a signal to travel over a 100-meter cable than it will a 10-meter cable. Deskew values are especially important when the logical ISL will cover long distances such as several miles. In such configurations, all physical cables forming the logical ISL need to have deskew values that meet your switch vendor's requirements. Not abiding by this rule will cause problems, such as the logical ISL not being able to form, or even worse, the logical ISL forming but performing poorly. The golden rule when it comes to ISL-related deskew values is *always follow your switch vendor's best practices!*

The following two commands show two physical ISLs formed into an ISL trunk between two Brocade switches. Each physical ISL is operating at 8 Gbps, creating a single logical ISL with 16 Gbps of bandwidth. Each physical ISL also has exactly the same deskew value—in our particular case, this is thanks to the fact that they are both running over short, 10-meter, multi-mode cables. It also shows that one of the physical ISLs in the trunk is the master. Not every vendor's implementation of logical ISLs has the concept of masters and subordinates, and having masters can be problematic if the master fails.

```
LegendarySw01:admin> islshow
  1: 14->14 10:00:00:05:AA:BB:CC:EE  03 LegendarySw03  sp:8G bw:16G TRUNK QOS

LegendarySw01:admin> trunkshow
  1:14->14 10:00:00:05:AA:BB:CC:EE    21 deskew 15 MASTER
    15->15 10:00:00:05:AA:BB:CC:FF    21 deskew 15
```

The following command shows a similar logical ISL formed from two physical ISLs on a Cisco MDS switch:

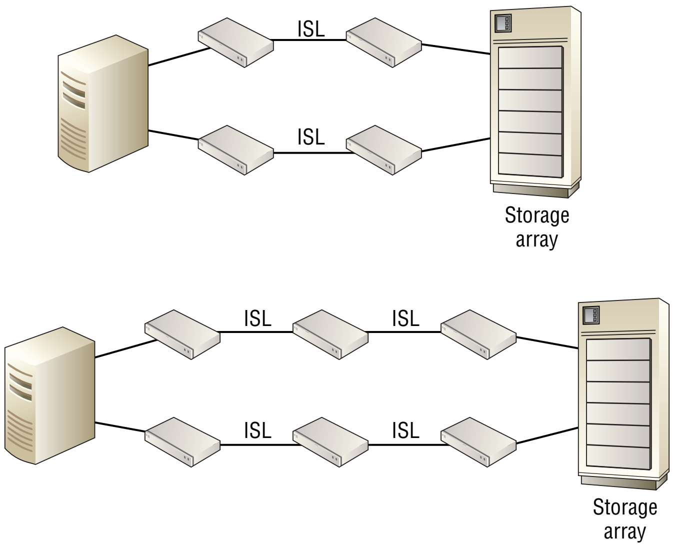
```
MDS-sw01# show port-channel database
port-channel 10
    Administrative channel mode is active
    Operational channel mode is active
    Last membership update succeeded
    2 ports in total, 2 ports up
    Ports:   fc2/1    [up]*
             fc2/8    [up]
```

One final consideration regarding ISLs is *hop count*. Each ISL that a frame has to traverse en route from source N_Port to destination N_Port is referred to as a *hop*. If your initiator and target are patched to separate switches that are directly connected to each other via a single ISL, any communications between the two will have to traverse the ISL, resulting in one hop. If two ISLs have to be traversed, that equates to two hops. It is a good practice to keep your SAN as flat as possible and keep hop count to a minimum. That said, nodes attached to the fabric have no concept of how many switches exist in a fabric.

Figure 5.15 shows two simple configurations, one with a single hop and the other with two hops required.

Within FC-switched fabrics, there are several loosely accepted topologies. By far the most popular is the core-edge topology. Other designs (such as cascade, ring, and mesh) do exist, but are rare compared to the almost ubiquitous core-edge. Let's have a close look at each.

FIGURE 5.15 Hop count



NOTE

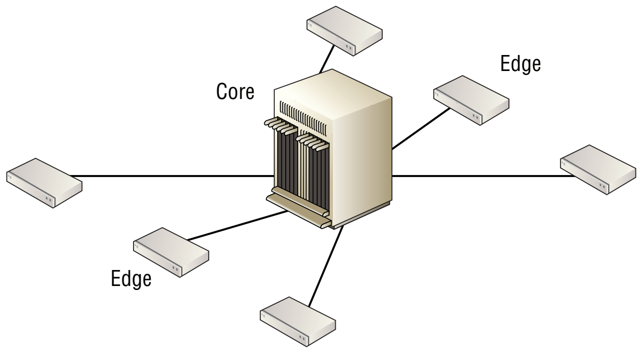
Because of the popularity of the core-edge design, and the relative lack of real-world implementations of other designs, one may be justified in thinking that the folks who came up with the alternative topologies were either desperate to imitate some of the already established Ethernet networking designs or simply had too much time on their hands and invented them to avoid boredom.

Before we look at some of the topologies, it is worth pointing out that when referring to a fabric topology, we are referring to switches and ISLs only. Moving an end device such as a host or storage array from one switch port to another, or even from one switch to another switch, does not usually affect the topology of the fabric. The notable exception is in a tiered topology.

Core-Edge Topology

The *core-edge topology* is by far the most widely implemented topology, and it will suit most requirements. A core-edge fabric looks a lot like an old hub-spoke, or a star network topology. You have a core switch at the physical center of the fabric, and each edge switch connects to the core switch via ISLs. Figure 5.16 shows a core-edge design with a single director at the core, and six edge switches connecting to the core.

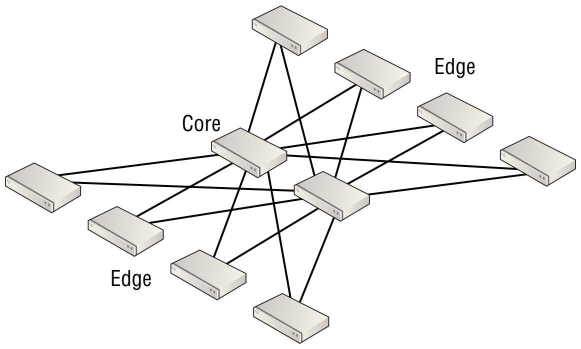
FIGURE 5.16 Core-edge topology



The core-edge design naturally lends itself to blade server environments whereby small SAN- or FCoE-capable switches are embedded in the blade server chassis. ISLs then run from these embedded-edge switches to a core switch, where the storage is directly connected.

It is perfectly acceptable for a core-edge design to have more than one core switch per fabric, as shown in Figure 5.17.

FIGURE 5.17 Core-edge design with two core switches



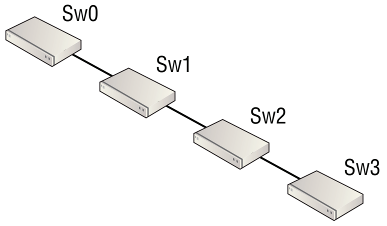
As far as fabric services are concerned, there is nothing special about the core switch. However, many people choose to force the core switch to be the principal switch in the fabric, as well as opting to perform most of the fabric management from the core switch. This does not have to be the case, though. However, having the core switch as the principal switch is a good idea because it is the switch least likely to be rebooted.

NOTE Although the FC core-edge design is similar to an Ethernet star network topology, the nature of FCP and its use of the FSPF routing protocol allows multiple core switches, and all the links between them, to be active at the same time. In contrast, Ethernet networks require Spanning Tree Protocol (STP) or Hot Standby Router Protocol (HSRP) to make certain paths standby paths in order to avoid loops.

Cascade Topology

Cascaded designs connect switches in a row, as shown in Figure 5.18.

FIGURE 5.18 Cascade topology



Cascade topology is an abomination, and 99.999 percent of the time should not be deployed, as it is not scalable, performant, or reliable. If switch 2 (Sw2) in Figure 5.18 fails, switch 3 (Sw3) is isolated from the rest of the fabric.

Ring Topology

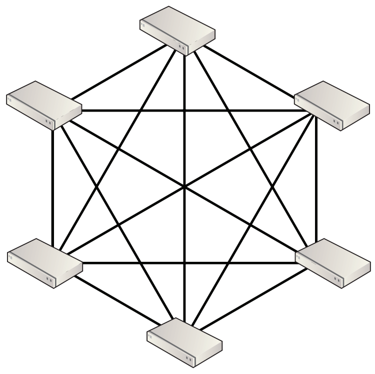
Ring topology is the same as cascade except that the two end switches are connected, making it ever so slightly less hideous than cascade topology. By linking the two end switches, you offer an alternative path to a destination by sending frames in the opposite direction around the ring. Obviously, if a switch in a ring design fails, the topology becomes a cascade, but devices do not become isolated, because of the alternative path. However, adding switches to a ring topology requires the ring to be broken.

Like cascade, it is not a popular design and has few if any practical use cases in the modern storage world.

Mesh Topology

Mesh topologies have every switch in a SAN connected to every other switch. This ensures that every switch is only a single hop away from any other switch and is highly resilient and performant. However, it is extremely costly from an ISL perspective. Basically, lots of switch ports get used up as ISLs. Figure 5.19 shows a simple mesh topology.

FIGURE 5.19 Mesh topology



Mesh topologies are increasingly wasteful of ports (because they need to be used as ISLs) as they scale.

Tiered Topologies

Tiered topologies are the enemy of localization. In a tiered fabric topology, you group all like devices on the same switch. For example, all hosts would go on a set of switches, all storage ports on a set of switches, all tape devices on a set of switches, and all replication ports on a set of switches. This design guarantees that hosts accessing storage or tape devices have to cross at least one ISL. This is not the end of the world, though, as long as you have sufficient ISL bandwidth.

Maintenance and expansion of tiered fabric designs is extremely simple compared to extreme localization. In a tiered design, it is relatively simple to increase bandwidth between tiers by simply adding ISLs. It is also easy to expand a tier by adding more switches or more switch ports to it.

In the real world, tiered fabric topologies are fairly common, especially if you consider placing all storage and tape devices on core switches and all hosts on edge switches in a hybrid *tiered core-edge topology*.

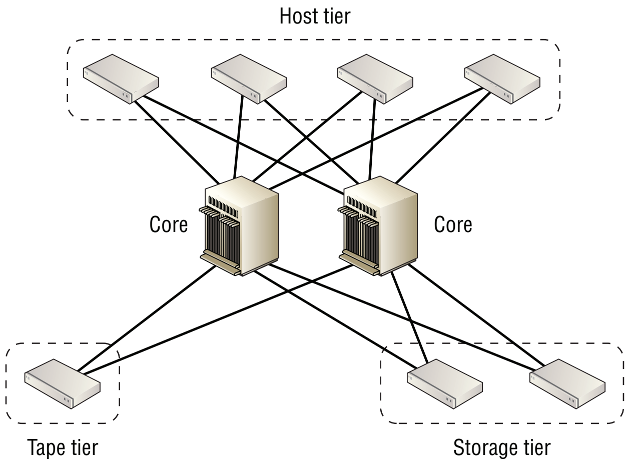
Figure 5.20 shows a simple core-edge fabric design that incorporates a form of tiering.

Despite all of this talk about topologies, it is not a requirement to design a fabric around any specific topology. You can mix and match and tweak a design to your requirements.

FC Routing

Now let's look at how frame delivery and routing works in FC fabrics, as well as some vendor implementations of virtual fabrics and virtual SANs.

FIGURE 5.20 Tiered fabric topology



Virtual Fabrics and VSANS

The two major FC switch vendors—Brocade and Cisco—have technologies that allow you to partition physical switches to the port level, allowing individual ports in a single physical switch to belong to discrete fabrics.

Cisco invented the technology and refers to it as virtual SAN (VSAN). The ANSI T11 committee liked it so much that they adopted it as an industry standard for virtual fabrics. Brocade then based its virtual fabrics implementation on the ANSI T11 standard. As a result, it shouldn't be much of a surprise that Cisco's VSAN and Brocade's virtual fabrics technologies are extremely similar.

Cisco VSAN

Each VSAN is a fully functioning fabric, with its own dedicated ports, devices, and fabric services. These fabric services include the usual suspects of the name server, zoning table, alias table, and so on.

All F_Ports on Cisco MDS and Nexus FCoE switches can be a member of one, and only one, VSAN. However, ISLs—including PortChannels—in *trunk mode* can carry traffic for multiple VSANs while maintaining isolation of traffic from the different VSANs. They do this by virtue of a VSAN tag that is added to the FC frame header to identify VSAN membership.

A good use case for VSANs might include partitioning the ports on a single switch into a development fabric and a staging fabric. This would allow you to have a single physical platform running two distinct, isolated, logical fabrics. You can dynamically move ports between the development and staging fabrics as required, allowing for good flexibility. If you mess up the zoning in the development fabric, the staging fabric will be unaffected. It works great, and you no longer need to buy dedicated hardware if you want to isolate fabrics!

While separating ports on isolated logical fabrics (VSANs), it is possible to route FC frames between devices on separate VSANs via a Cisco technology referred to as *inter-VSAN routing (IVR)*. IVR works by creating zones and adding them to the IVR zone set. Zones in the IVR zone set effectively overlap VSANs, allowing devices that are in separate VSANs but exist in the IVR zone set to communicate with each other.

Using IVR Properly

A mistake that is occasionally made by customers overly keen to deploy VSAN technology with IVR is to place all storage arrays in one VSAN and all of their hosts in another VSAN in a form of tiered virtual fabric topology, and then use IVR to enable the hosts to communicate across VSANs with the storage arrays. This is not the best use case for VSANs and could result in IVR becoming a choking point.

This can be easily resolved by placing storage ports and host ports in the same VSAN. It should also be noted that this problem is not a flaw in VSAN or IVR technology, but rather an example of the technologies being used inappropriately. IVR is a great technology, but one that is best used for exceptions rather than the rule.

Brocade Virtual Fabrics

Brocade virtual fabrics are essentially the same as Cisco VSANs. They allow you to carve up physical switches by allocating ports to newly created logical switches. Multiple logical switches then form logical fabrics, each of which has its own separate fabric services—name server, zoning, domain IDs, principal switches, and the like.

Use cases are the same as with Cisco VSANs. Take a single physical piece of tin, carve its resources into discrete logical entities that can be managed separately, and maintain traffic isolation between the logical entities. Again, mess up the zoning DB on a Brocade logical fabric, and all other logical fabrics remain unaffected.

As with Cisco VSAN technology, it is possible to route traffic between logical fabrics by using Brocade's Fibre Channel Routing (FCR) technology and Logical Storage Area Network (LSAN) zones. However, the Brocade FCR technology requires a special base switch to act as a backbone fabric that enables routing between two separate fabrics.

WARNING

Cisco IVR and Brocade FCR should be used sparingly. Use either of them too much, and you risk giving yourself a major headache. Both technologies are great, but make sure you use them appropriately and don't abuse them.

FC Naming

Every end device on an FC SAN has two important identifiers:

- World wide name
- Node port ID

As a storage guy, you will interact with world wide names and world wide port names more often than you will interact with N_Port IDs.

World Wide Names

Each node port—host HBA/CNA port, storage array port—has its own 64-bit (8-byte) worldwide unique number that stays with its device for its entire life. This 64-bit name is known as a *world wide name (WWN)*. Actually, each node port has a world wide node name (WWNN) and a world wide port name (WWPN). More often than not, you will work with WWPNs.

NOTE

Nodes, in the sense of servers and storage arrays, do not have addresses. It is the ports in the servers and storage arrays that have addresses. This means that you cannot address a server or storage array; you can address only the ports in the server or storage array. While this may appear to be effectively the same thing, it means that if a storage array has 64 ports, you cannot address all of the 64 ports by specifying a single address that all 64 ports on the array respond to.

The world wide port name is sometimes referred to as a port world wide name, or pWWN. This book uses the term WWPN.

The following output from a `switchshow` command on a Brocade FC switch shows the switch's WWNN as well as the WWPNs of the end devices connected to the first two FC ports in the switch (as mentioned, you cannot address traffic to the switch by its WWN):

```
LegendarySw01:admin> switchshow
SwitchNAme:   LegendarySw01
SwitchType:   62.3
SwitchState:  Online
SwitchMode:   Native
SwitchRole:   Principal
SwitchDomain: 44
SwitchId:     fffXXX
SwitchWwn:    10:00:00:05:AA:BB:CC:DD
Zoning:       ON (Prd_Even1)
SwitchBeacon: OFF

Index Port Address Media Speed State     Proto
```

```
==============================================
  0   0    220000  cu    8G    Online    FC  F_Port 50:01:43:80:be:b4:08:62
  1   1    220100  cu    8G    Online    FC  F_Port 50:01:43:80:05:6c:22:ae
```

The following nsshow command shows the WWNN and WWPN of a host HBA:

```
LegendarySw01:admin> nsshow
Type Pid    COS     PortName                NodeName                 TTL(sec)
N    220100;  3;    50:01:43:80:05:6c:22:ae;50:01:43:80:05:6c:22:af   na
    FC4s: FCP
    NodeSymb: [43] "QMH2462 FW:v5.06.03 DVR:v8.03.07.09.05.08-k"
    Fabric Port Name: 10:00:00:05:AA:BB:CC:DD
    Permanent Port Name: 50:01:43:80:05:6c:22:af
    Port Index: 1
    Share Area: No
    Device Shared in Other AD: No
    Redirect: No
    Partial: No
```

WWNs—both WWNN and WWPN—are manufacturer assigned. WWNs are written as 16 hexadecimal digits, with every two digits being separated by a colon. WWNs can contain only hexadecimal characters and are therefore not case sensitive, meaning that 50:01:43:80:05:6c:22:af and 50:01:43:80:05:6C:22:AF are identical.

WWNs are burned into the HBA/CNA card, similar to the way a MAC address is burned into a NIC. However, that should be where the MAC address analogy stops! Unlike MAC addresses, WWNs are not used for transporting frames on the fabric. It is the 24-bit N_Port ID (covered later in this chapter) that is used for frame routing and switching.

WWNs are used primarily for security-related actions such as zoning and other device security, which are also covered in detail later.

NOTE

Although WWNs are assigned to HBA and CNA cards at the factory, they can usually be changed via the management tools provided with them. This can be particularly useful if you're deploying WWN-based zoning, as rather than update your aliases and the like, you can simply force any new replacement HBA or CNA to have the same WWN as the HBA or CNA it is replacing.

N_Port IDs

As we mentioned earlier, WWNs are not used for frame switching and delivery on FC networks. This is where N_Port IDs come into play.

NOTE Although N_Port IDs (FCIDs) are vital to switching and routing, SAN administrators rarely have to interact with them.

Before exploring an N_Port ID, let's look at what an N_Port is. *N_Port* is FC lingo for *node port*, and it refers to physical FC ports in end devices on FC networks. Basically, all storage array ports and host HBA ports are end points and therefore are considered N_ Ports. FC switches, on the other hand, do not act as end points and therefore do not have N_Ports. N_Ports apply to only FC-P2P and FC-SW architectures. But these are just about the only topologies being deployed these days anyway.

Some people refer to N_Port IDs as Fibre Channel IDs, or FCIDs for short. You can use either term. This book uses N_Port ID.

An N_Port ID is a 24-bit dynamically assigned address. It is assigned by the fabric when an N_Port logs in to the fabric. Every end point in the fabric has one, and it is this N_Port ID that is coded into the FC frame header and used for frame switching, routing, and flow control.

Let's take another look at a `switchshow` output and highlight the N_Port ID of the connected devices in bold:

```
LegendarySw01:admin> switchshow
SwitchNAme:   LegendarySw01
SwitchType:   72.3
SwitchState:  Online
SwitchMode:   Native
SwitchRole:   Principal
SwitchDomain: 44
SwitchId:     fffXXX
SwitchWwn:    10:00:00:05:AA:BB:CC:DD
Zoning:       ON (Prd_Even1)
SwitchBeacon: OFF

Index Port Address Media Speed State     Proto
==============================================
  0   0    220000  cu    8G    Online    FC  F_Port 50:01:43:80:be:b4:08:62
  1   1    220100  cu    8G    Online    FC  F_Port 50:01:43:80:05:6c:22:ae
```

N_Port IDs are smaller than WWPNs and are included in every frame header, which allows for smaller frame headers than if WWNs were used for frame routing and end-to-end delivery.

Also, just to be clear, a device with two physical ports connected to a SAN will get two N_Port IDs, one for each port.

The format and structure of N_Port IDs is as follows:

Domain The first 8 bits of the address are known as the domain ID. This can be a value from 1–239, and it is the domain ID of the switch the port belongs to.

Area This is the next 8 bits, and the switch vendor assigns and structures this part of the address. The value in this section of the N_Port ID can be anything from 0–255.

Port The last 8 bits of the address are also vendor specific and usually translate to the port number of the port in the switch. The value can be anything from 0–255, and switches with more than 256 ports use a combination of area and port to uniquely identify the port.

This hierarchical nature of N_Port IDs assists with frame routing and delivery. It is only the domain portion of the N_Port ID that is used in routing decisions until the frame reaches the destination switch.

NOTE One of the reasons WWNs were designed to be persistent and never change is precisely because N_Port IDs will change if a switch gets a new domain ID.

N_Port ID Virtualization

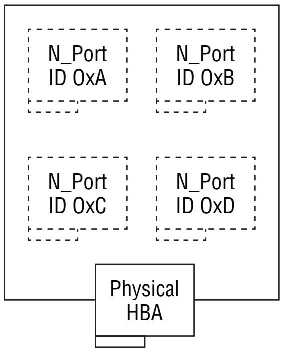
Originally, a physical N_Port logged in to a fabric and got a single N_Port ID back from the fabric login server. Things were nice and simple. However, with the emergence of server virtualization technologies, it became desirable for a physical N_Port to be able to register multiple N_Ports and get back multiple N_Port IDs so that an N_Port ID could be assigned to a virtual machine, and the SAN zoning could limit access to some resources to that VM alone. Witness the birth of N_Port ID virtualization (NPIV).

Before the days of NPIV, an N_Port would perform a FLOGI to obtain an N_Port ID from the fabric and then perform a PLOGI to register its N_Port ID with the name server. Job done. Fast forward to today, and a node supporting NPIV registers multiple virtual N_Ports with the fabric. Registration of additional N_Ports on the fabric is not done via more FLOGI operations, though. It is done via FDISC operations. Basically, an N_Port can perform as many FDISC operations as it wishes, with each FDISC followed by a PLOGI for the acquired N_Port ID. Figure 5.21 shows a single host with a single physical HBA that has registered an additional four N_Port IDs with the fabric.

In order to get additional N_Port IDs for each virtual N_Port, the FDISC operation registers an additional WWPN with the fabric. This means that NPIV-capable devices have to have multiple WWPNs—one for each N_Port ID it wants. Also, in order for NPIV to work, the fabric and switches obviously have to support NPIV. Otherwise, they will get confused when they see multiple N_Port IDs and WWPNs originating from a single host connected to a single F_Port.

NOTE Registration for registered state-change notifications (RSCN) is per WWPN/N_Port ID. Registration is also performed after each FLOGI and PLOGI.

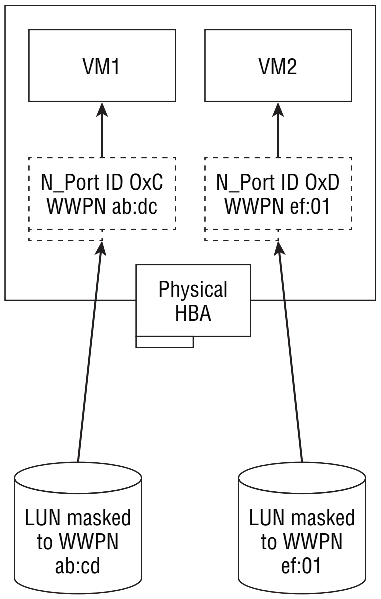
FIGURE 5.21 Single physical N_Port with four virtual N_Port IDs



Generally speaking, on the HBA/CNA side of things, NPIV-aware HBAs are capable of carving themselves up into multiple virtual HBAs. Each virtual HBA has its own WWPN and can therefore register for its own N_Port ID as if it were a fully fledged physical N_Port. This allows each virtual HBA to be directly addressed on the fabric. Couple this with the ability to associate a virtual HBA with a VM, and bingo, you have a VM that is directly addressable on the fabric and can have storage presented directly to it.

Figure 5.22 shows two LUNs presented directly to the WWPNs that are associated with two VMs in a single server behind a single physical HBA.

FIGURE 5.22 NPIV and presenting LUNs directly to VMs



Of course, mapping LUNs directly to VMs and bypassing the hypervisor is not universally liked, nor is it appropriate for all situations. But it definitely has its place.

NPV and Access Gateway Modes

First up, NPV is not the same as NPIV. It looks the same, and it's based on NPIV, but it's not the same thing!

NPV is an abbreviation for *N_Port Virtualizer*, and as a technology, the main goal of NPV is to enable switches from multiple vendors to coexist in the same fabric without having to resort to nasty interop modes. NPV also allows you to have lots of switches in a fabric without overloading the fabric services and depleting the finite number of supported switches—actually, the finite number of domain IDs—in a fabric.

NOTE Cisco invented NPV, and in true vendor vs. vendor style, Brocade and QLogic then implemented the same technology but had to call it something different. Brocade calls it *Access Gateway mode* (AG). QLogic refers to it as *intelligent pass-thru.*

It is time for a bit of background as to why NPV is so cool.

When an FC switch joins a fabric, it gets its own unique Domain_ID. It also participates in a load of fabric services, some of which we have discussed. As you add more and more switches to a fabric, you start to approach scalability limitations such as maxing out the number of supported domain IDs in a fabric. NPV helps here. We'll explain how in a second.

Also, if you build a fabric by using switches from Cisco, Brocade, and QLogic, you are going to have to get down with some funky interop mode madness and end up with a fabric with a very limited feature-set that is going to be a support headache from day one. NPV comes to the rescue here too.

So, the secret behind NPV is that a switch in NPV mode is invisible to the fabric. It doesn't register for a domain ID and it doesn't share the burden of assisting with fabric services. As far as the fabric is concerned, the switch is not a switch; it just looks like an N_Port running NPIV (yes, NPIV, not NPV).

A common use case is running NPV switches at the edge and leaving the core switches to take care of all of the fabric services. This is popular in situations where customers are adding Cisco UCS blade servers to environments that already operate Brocade or QLogic fabrics. Edge to core appears to be the sweet spot for NPV, and you wouldn't want to have to use it between core switches over a long distance.

Interestingly, the links between switches running NPV are not ISLs, and the ports are not E_Ports. The ports on the edge switch are sometimes referred to as NP_Ports, and the ports on the core switches are just normal F_Ports.

Also of interest is that the NPV switches on the edge effectively proxy all of the FLOGI and PLOGI operations from the end devices connected to them. The NPV switch on the edge performs a FLOGI and PLOGI to the core switch for itself. It then converts FLOGIs from connected hosts to FDISCs and proxy registers each end device with the core as per NPIV. This way, the core switch just thinks it's talking to an NPIV node over a standard F_Port. It is simple and effective.

NPV is a standards-based technology, which is better than interop modes ever were. The result is good support between vendors. Interop mode is dead; long live NPV!

There are a couple of things to be aware of. Load balancing and HA of links from edge to core can vary between vendors. Load balancing tends to be basic round-robin, but this may change in the future. Also, be sure to check with your vendors about support of VSAN and virtual fabric technologies, especially trunking VSANs over connections between edge and core switches operating in an NPV configuration.

To see whether NPV/AG mode is enabled on a switch, issue the following commands:

```
LegendarySwitch01:admin> ag --modeshow
Access Gateway mode is enabled
```

The following output on a Cisco MDS fabric shows two switches that are configured in NPV mode in VSAN 8:

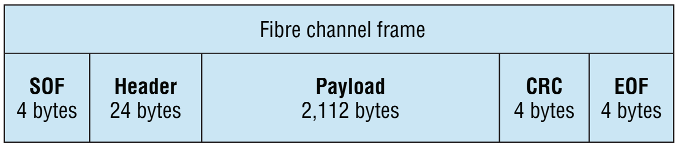
```
MDS-sw01# show fcns database
VSAN 8:
--------------------------------------------------------------------------
FCID TYPE PWWN (VENDOR) FC4-TYPE:FEATURE
--------------------------------------------------------------------------
0x010000 N 20:01:00:0d:ec:AA:BB:CC (Cisco) npv
0x010001 N 20:02:00:0d:ec:AA:BB:CD (Cisco) npv
```

Framing

Fibre Channel is essentially a unicast frame-based protocol. Most of what goes on in an FC SAN can be viewed as traditional layer 2 networking. FC frames can be up to 2,148 bytes long and can contain a payload of up to 2,112 bytes that must be divisible by 4.

Figure 5.23 shows an FC frame.

FIGURE 5.23 FC frame



Now let's explain each component of the frame, starting at the left and working our way to the right.

Start of Frame All FC frames are prefaced with a start of frame (SOF) ordered set that acts as a form of frame delimiter. The SOF is a single 4-byte word indicating the class of service (COS) and at what point this particular frame figures in a set.

Header Directly following the SOF is a 24-byte frame header. This contains the source and destination N_Port IDs, referred to as source ID (SID) and destination ID (DID). The header also defines whether the frame contains user data (data frame) or control data (link control frame) and indicates where a frame is featured in an exchange or sequence.

Payload Following the header comes the payload. This can be up to 2,112 bytes long and must be padded if not divisible by 4.

Cyclic Redundancy Check Frame integrity is possible via a cyclic redundancy check (CRC).

End of Frame Last but not least comes the end of frame (EOF) ordered set. This informs the receiving node that the frame is ended, and the specifics of the EOF are determined by the class of service of the frame.

The preceding is good theory, but it is unlikely that you will need to call upon such knowledge very often in your day-to-day storage administration.

Flow Control

First you need to know a bit of background before we jump into the specifics of flow control. At a fairly high level, communication on an FC SAN is composed of three basic constructs: frames, sequences, and exchanges. The smallest and most fundamental unit is the *frame*. Multiple related frames form a *sequence*, and multiple related sequences form an *exchange*. A conversation is frequently used as an analogy. In this analogy, frames are words, sequences are sentences, and exchanges are conversations. That is good to know, but you probably won't rely on that knowledge every day in your job.

Flow control in an FC SAN is based on a credit system. End devices are issued with credits and are allowed to send one frame for every credit that they have. If there are no credits, no frames can be sent! This approach works fairly well for avoiding congestion and frame dropping.

There are two major credit systems in use:

- Buffer to buffer (B2B)
- End to end (E2E)

B2B credit works between two directly connected ports such as an end device and a switch port (N_Port to F_Port). E2E credit works between two end devices such as an initiator and a target.

End-to-End vs. Buffer-to-Buffer Flow Control

E2E flow control (sometimes referred to as EE_credit) works on the principle of the destination device returning an Accept (ACC) to the sender to indicate that it has received a frame and that the sender can increment its credit stock by one. Each time the sender sends a frame, it decreases its credit stock by one, and each time it receives an ACC, it increments its credit stock by one. This ACC-based replenishing scheme works with class of service 1 and class of service 2 because they both work on the principle of end-to-end connectivity and use acknowledgments. These acknowledgments allow the sender to maintain a rolling stock of credits. Switches have no involvement in E2E credit schemes.

NOTE The different classes of service are discussed in more detail in the upcoming "Classes of Service" section of this chapter. However, most FC traffic, including disk and tape I/O, uses COS 3.

B2B flow control (sometimes called BB_credit) works between node ports and fabric ports. As B2B is used by COS 3, which is connectionless without acknowledgments, ACC frames are not used to replenish credits. Instead, a special receiver ready frame (R_RDY) is issued back to the sender to indicate that the recipient is ready to receive more frames. Similar to E2E, in a B2B credit system, the sender decreases its credit stock by one for every frame it sends and increases it by one for every R_RDY it receives.

Both E2E and B2B credit systems require the sender to have an initial stock of credits. This credit stock is agreed at FLOGI for B2B and at PLOGI for E2E. This initial credit stock is important because if a device started life in a fabric without any credits, it would never be able to send frames.

Since we're on the topic, I suppose there is no harm in mentioning that arbitrated loop (FC-AL) environments support a form of B2B credit system. However, in this situation, each device works with an initial credit of zero. Once an initiator arbitrates control of the wire, it contacts the target, which responds with a number of R_RDYs representing the number of credits it wishes the initiator to have.

Error Recovery

FC frame recovery works at the sequence level, meaning that if a frame is dropped or lost, the entire sequence must be re-sent. This isn't the end of the world, because most FC SANs are stable and not congested, meaning that recovery isn't required very often.

Flow Control over Long Distances

In stretched or extended fabric configurations, switch ports at either end of the long-distance link are configured with extra buffers to allow more frames to be sent and exist on the link without having to wait for acknowledgments.

Generally speaking, the longer the distance between the sending and receiving port, the more credits are assigned to the port. If additional credits weren't assigned, the effective bandwidth of these links would be reduced as the sender waited for R_RDY frames, due to the bandwidth-delay product. Also, data can be sent on the transmit line and credits can be replenished on the receive line in full-duplex mode.

Classes of Service

In an attempt to provide differing levels of service—basically, delivery guarantees and bandwidth consumption—the FC Protocol defines seven classes of service (COS). However, not only are they not all implemented, the vast majority of FC traffic is either Class 3 or Class F, with a smattering of Class 2 occasionally seen in the real world.

Class 1

Class 1 is a connection-oriented link that uses acknowledgments. While it isn't widely implemented, you may see it on an exam. In Class 1 connections, the initiator and target consume the full bandwidth of the link and maintain a virtual circuit until the transmission sequence ends. Acknowledgment frames are used to ensure reliable and in-order frame delivery. Class 1 connections use only E2E flow control.

Class 2

Class 2 is a connectionless link that uses acknowledgments. It's kind of similar to TCP, if you know your TCP/IP theory. Basically, it is not a dedicated connection but requires delivery acknowledgments. Class 2 connections can request in-order frame delivery, but it is not mandatory. Out-of-order frame delivery in FC SANs is rare compared to Ethernet networks.

Class 2 connections use both E2E and B2B flow control. Class 2 is seen more often than class 1, but still not very often.

Class 3

Class 3 is a connectionless service that does not use acknowledgment frames. This is easily the most common class of service seen in FC fabrics and is conceptually similar to UDP in the TCP/IP world. If you know your TCP/IP theory, due to its similarity with UDP, COS 3 is sometimes referred to as a datagram service.

Generally speaking, FC SAN environments do not drop or discard frames, meaning that acknowledgment frames are an unnecessary overhead—unless, of course, you utilize these ACC frames to replenish credits. If frames are dropped, it is the responsibility of the upper-layer protocols to clean up the mess and ensure retransmission.

Class 3 uses only B2B flow control.

Class 4

Class 4 defines a connection-based service with acknowledgments. It differs from class 1 in that it consumes fractional bandwidth and can apply quality of service to connections. However, it is rarely implemented.

Class F

Class F is a lot like class 2. It is connectionless but utilizes acknowledgment frames. Class F is the de facto COS used for switch-to-switch communications. It operates only across E_Ports and ISLs. It is used for delivery of control data such as state-change notifications, name server updates, routing info, zoning, build fabric requests, and reconfigure fabric requests. Class F uses the B2B flow control mechanism.

Oversubscription

Oversubscription occurs when more than one device utilizes a single link. ISLs are often oversubscribed. For example, an edge switch with 24 8 Gbps ports may be configured as follows:

- 22 hosts each connected at 8 Gbps
- 2 ISL ports configured as a single 16 Gbps ISL to the core switch

In order for the 22 hosts to access the storage array connected to the core switch, all 22 hosts have to cross the 16 Gbps ISL. If all 22 hosts utilized their full 8 Gbps, the ISL would be a severe bottleneck. However, if you know that the combined throughput of all your hosts will never push anywhere near the maximum theoretical throughput of their 8 Gbps, you can safely oversubscribe your ISL as described.

If all ports in a fabric operate at the same speed, oversubscription is simply the ratio of node ports to ISLs. So if you have 10 hosts at 8 Gbps, and a single ISL at 8 Gbps, you have an oversubscription ratio of 10:1. However, in the previous example, we have a logical ISL operating at 16 Gbps and 22 hosts operating at a theoretical 8 Gbps each, so our oversubscription ratio would be 11:1.

Oversubscription isn't just used for ISLs. It is also common for a single storage port to be zoned with multiple hosts in a fan-out configuration. For example, a single storage port might be zoned with 10 host ports in a 1:10 fan-out ratio. If the hosts and storage port are operating at the same speed, we have a 10:1 oversubscription ratio. Again, this is extremely common in the real world because 99 percent of the time hosts do not, and cannot, drive their full bandwidth. This is especially so when you consider that most hosts will have multiple active links (for redundancy).

Locality

Locality is the degree to which communication between two devices is kept as close as possible. For example, two communicating devices connected to the same switch have a high degree of locality. In contrast, two devices that are on separate switches three hops apart have a very low degree of locality.

Locality can help avoid congestion by sticking storage and hosts close to each other, but don't get carried away with it. It is perfectly acceptable to have hosts attached to edge switches, and storage attached to core switches. This is common in blade server configurations, where there are embedded edge switches in the blade server chassis.

NOTE

In most FC fabrics, switch-based latency is not worth losing any sleep over. It is often in the region of a miserly ~1 microsecond per switch due to the high-speed cut-through switching mechanisms that many FC switches use. Any normal fabric-related latency such as frame switching or ISL hopping is miniscule when compared with the latency of reading data from spinning disk. And, of course, solid-state technologies such as flash media avoid many of the latencies associated with old spinning disk media.

Real World Scenario

Be Careful Not to Over-Engineer

Be careful not to get too carried away with locality, as you can easily overcomplicate your design for little or no benefit.

One company went to the extreme of taking storage ports and host ports that were zoned to each other and connecting them to adjacent switch ports that shared a common switching ASIC. The idea was that traffic between the hosts and storage would be *switched* at extremely low latency on the ASIC and never have to incur the latency associated with traversing the switches' backplane, never mind any latency associated with crossing ISLs.

While the design looked good on paper, it was over-engineered and created an unsustainable configuration that was cumbersome and too difficult to maintain and ultimately didn't scale. Lots of hard work was done for probably no realizable benefit. Simplicity is king!

Fabric Reconfigurations

Certain fabric-related events can cause a fabric to reconfigure. Some of these events include the following:

- New switches added to the fabric.
- Merging fabrics.
- A new principal switch is selected.

There are two types of fabric reconfigure events:

- Build fabric (BF)
- Reconfigure fabric (RCF)

Build fabric events are nondisruptive, but reconfigure fabric events are disruptive.

During an RCF event, a principal switch election is held, domain IDs are assigned, and routes are established. It is entirely possible that the same principal switch will be reelected, and that domain IDs will not change. However, if they do change, this can be a pain in the neck, especially if you are using port zoning.

For these reasons, it is obviously preferred to perform a BF rather than an RCF.

FC SAN Troubleshooting

As you have seen, there are a lot of ways you can handle FC SAN. Consequently, there are a lot of areas where you can experience trouble with your configurations. Let's take a look at some of the ways you can troubleshoot FC SAN.

Backing Up Your Switch Configurations

While not exactly troubleshooting, you should make regular backups of your switch configurations, just in case you are ever in a situation where things are fouled up beyond all repair (FUBAR) and you have to revert to a previous configuration. Such backup files tend to be human-readable flat files that are extremely useful if you need to compare a broken configuration image to a previously known working configuration. Another option might be to create a new zone configuration each time you make a change, and maintain previous versions that can be rolled back to if there are problems after committing the change.

Troubleshooting Connectivity Issues

Many of the day-to-day issues that you will find yourself troubleshooting include connectivity issues such as hosts not being able to see a new LUN or not being able to see storage or tape devices on the SAN. More often than not, connectivity issues will be due to misconfigured zoning.

Each vendor provides different tools to configure and troubleshoot zoning, but the following common CLI commands can prove very helpful.

fcping

fcping is an FC version of the popular IP ping tool. fcping allows you to test the following:

- Whether a device (N_Port) is alive and responding to FC frames
- End-to-end connectivity between two N_Ports
- Latency
- Zoning (some implementations check zoning between two devices)

fcping is available on most switch platforms as well as being a CLI tool for most operating systems and some HBAs. It works by sending Extended Link Service (ELS) echo request frames to a destination, and the destination responding with ELS echo response frames.

The following fcping command is run on a Brocade switch and proves that a device is alive and responding to FC frames:

```
LegendarySw01:admin> fcping 50:01:43:80:05:6c:22:ae
Pinging 50:01:43:80:05:6c:22:ae [0x 220100] with 12 bytes of data:
received reply from 50:01:43:80:05:6c:22:ae: 12 bytes time:1162 usec
received reply from 50:01:43:80:05:6c:22:ae: 12 bytes time:1013 usec
received reply from 50:01:43:80:05:6c:22:ae: 12 bytes time:1442 usec
received reply from 50:01:43:80:05:6c:22:ae: 12 bytes time:1052 usec
received reply from 50:01:43:80:05:6c:22:ae: 12 bytes time:1012 usec
5 frames sent, 5 frames received, 0 frames rejected, 0 frames timeout
Round-trip min/avg/max = 1012/1136/1442 usec
```

The following fcping command is run to the same WWPN when the device is not alive on the network:

```
LegendarySw01:admin> fcping 50:01:43:80:05:6c:22:ae
Pinging 50:01:43:80:05:6c:22:ae [0x 220100] with 12 bytes of data:
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
5 frames sent, 0 frames received, 0 frames rejected,5 frames timeout
Round-trip min/avg/max = 0/0/0 usec
```

The following command shows the same host being fcpinged, but this time from a Linux host on interface n0.170 and specifying the N_Port ID instead of the WWPN:

```
fcping -c 3 -h n0.170 -F 0x220100
sending echo to 0x220100
echo    1 accepted                      16.170 ms
echo    2 accepted                       7.053 ms
echo    3 accepted                       6.803 ms
3 frames sent, 3 received 0 errors, 0.000% loss, avg. rt time 10.009 ms
```

Some switch vendors also check the zoning table when you specify two N_Port IDs in the same fcping command. For example, the following fcping command on a Brocade switch checks the zoning table to ensure that the two WWPNs specified are correctly zoned to each other:

```
LegendarySw01:admin> fcping 50:01:43:80:be:b4:08:62 50:01:43:80:05:6c:22:ae
Source: 50:01:43:80:be:b4:08:62
```

```
Destination: 50:01:43:80:05:6c:22:ae
Zone Check: Not Zoned
```

fctrace

Another tool that is modeled on a popular IP networking tool is the *fctrace* tool. This tool traces a route/path to an N_Port. The following command shows an fctrace command on a Cisco MDS switch:

```
MDS-sw01# fctrace fcid 0xef0010 vsan 1
Route present for :  0xef0010
20:00:00:05:30:00:59:ba(0xfffbee)
Latency: 0 msec
20:00:00:05:30:00:58:1b(0xfffc6b)
Timestamp Invalid.
20:00:00:05:30:00:59:1e(0xfffcef)
Latency: 174860 msec
20:00:00:05:30:00:58:1b(0xfffc6b)
```

Brocade has a similar tool called pathinfo.

While on the topic of zoning, these are some good things to consider:

- Are your aliases correct?
- If using port zoning, have your switch domain IDs changed?
- If using WWPN zoning, have any of the HBA/WWPNs been changed?
- Is your zone in the active zone set?

Rescanning the SCSI Bus

After making zoning changes, LUN masking changes, and pretty much any other work that changes a LUN/volume presentation to a host, you may be required to rescan the SCSI bus on that host in order to detect the new device. The following command shows how to rescan the SCSI bus on a Windows server using the `diskpart` tool:

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online           68 GB      0 B
  Disk 1    Online          140 GB  1024 KB
  Disk 2    Online          100 GB  1024 KB
```

This output shows three volumes. However, we have just presented a fourth volume to this host that is not showing up. To rescan the SCSI bus, we will issue a `rescan` command:

```
DISKPART> rescan

Please wait while DiskPart scans your configuration...

DiskPart has finished scanning your configuration.
```

Now that we have issued the rescan, the following `list disk` command now shows all four devices:

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online           68 GB      0 B
  Disk 1    Online          140 GB  1024 KB
  Disk 2    Online          100 GB  1024 KB
  Disk 3    Online          250 GB   250 GB
```

If you know that your LUN masking and zoning are correct but the server still does not see the fourth device, it may be necessary to reboot the host.

Understanding Switch Configuration Dumps

Each switch vendor also tends to have a built-in command/script that is used to gather configs and logs to be sent to the vendor for their tech support groups to analyze. The output of these commands/scripts can also be useful to you as a storage administrator. Each vendor has its own version of these commands/scripts:

Cisco—`show tech-support`

Brocade—`supportshow` or `supportsave`

QLogic—`create support`

Let's look at a cut-down example of a `show tech-support brief` on a Cisco MDS switch. From the truncated output, you can see that the switch has two configured VSANs and lists two ports, one configured in each VSAN. You can also see other interesting information, such as the management IP address, firmware version, and so on:

```
MDS-sw01# show tech-support brief
Switch Name          : MDS-sw01
Switch Type          : DS-X9216-K9
Kickstart Image      : 1.3(2) bootflash:///m9200-ek9-kickstart-mz.1.3.1.10.bin
System Image         : 1.3(2) bootflash:///m9200-ek9-mz.1.3.1.10.bin
```

```
IP Address/Mask       : 10.10.10.101/24
Switch WWN            : 20:00:00:05:30:00:75:32
No of VSANs           : 2
Configured VSANs      : 1,9

VSAN    1:    name:VSAN0001, state:active, interop mode:default
              domain id:0x6d(109), WWN:20:01:00:05:30:00:84:9f [Principal]
              active-zone:VR, default-zone:deny

-------------------------------------------------------------------------------
Interface  Vsan   Admin  Admin   Status          FCOT   Oper  Oper   Port
                  Mode   Trunk                          Mode  Speed  Channel
                         Mode                                 (GBps)
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Interface  Vsan   Admin  Admin   Status          FCOT   Oper  Oper   Port
                  Mode   Trunk                          Mode  Speed  Channel
                         Mode                                 (GBps)
-------------------------------------------------------------------------------


fc1/1      1      auto   on      fcotAbsent       --     --            --
fc1/2      9      auto   on      fcotAbsent       --     --            --

<output truncated>
```

Using Port Error Counters

Switch-based port error counters are an excellent way to identify physical connectivity issues such as these:

- Bad cables (bent, kinked, or otherwise damaged cables)
- Bad connectors (dust on the connectors, loose connectors)

The following example shows the error counters for a physical switch port on a Brocade switch:

```
LegendarySw01:admin> portshow 4/15
portName: LegendaryHost_HBA1
portHealth: HEALTHY

......
portFlags: 0x24b03       PRESENT ACTIVE F_PORT G_PORT U_PORT LOGICAL_ONLI
```

```
portType:  4.1
portState: 1    Online
portPhys:  6    In_Sync
portScn:   32   F_Port

Distance:  normal
portSpeed: 8GBps

Interrupts:        90352      Link_failure: 44         Frjt:         0
Unknown:           32         Loss_of_sync: 3          Fbsy:         0
Lli:               5240       Loss_of_sig:  11
Proc_rqrd:         84810      Protocol_err: 0
Timed_out:         276        Invalid_word: 0
Rx_flushed:        0          Invalid_crc:  0
Tx_unavail:        10         Delim_err:    0
Free_buffer:       2          Address_err:  0
Overrun:           0          Lr_in:        18
Suspended:         0          Lr_out:       50
Parity_err:        0          Ols_in:       45
2_parity_err:      0          Ols_out:      33
CMI_bus_err:       0
```

These port counters can sometimes be misleading. It is perfectly normal to see high counts against some of the values, and it is common to see values increase when a server is rebooted and when similar changes occur. If you are not sure what to look for, consult your switch documentation, but also compare the counters to some of your known good ports. If some counters are increasing on a given port that you are concerned with, but they are not increasing on some known good ports, then you know that you have a problem on that port.

Other commands show similar error counters as well as port throughput. The following `porterrshow` command shows some encoding out (`enc out`) and class 3 discard (`disc c3`) errors on port 0. This may indicate a bad cable, a bad port, or another hardware problem:

```
LegendarySw01:admin> porterrshow
        frames    enc  crc  too  too  bad  enc  disc link   loss   loss
       tx     rx  in   err  shrt long eof  out   c3  fail    sync   sig
=========================================================================
0:    1.4g   4.2g 0    0     0    0    0    9    183  0       0      0
1:    0      0    0    0     0    0    0    0      0  0       0      0
2:    1.0g   2.4g 0    0     0    0    0    0      0  0       0      0
3:    0      0    0    0     0    0    0    0      0  0       0      0
4:    0      0    0    0     0    0    0    0      0  0       0      0
```

The following portstasshow command shows a clean port with no errors registered:

```
LegendarySw01:admin> portstatsshow 1/6
stat_wtx                0           4-byte words transmitted
stat_wrx                0           4-byte words received
stat_ftx                0           Frames transmitted
stat_frx                0           Frames received
stat_c2_frx             0           Class 2 frames received
stat_c3_frx             0           Class 3 frames received
stat_lc_rx              0           Link control frames received
stat_mc_rx              0           Multicast frames received
stat_mc_to              0           Multicast timeouts
stat_mc_tx              0           Multicast frames transmitted
tim_rdy_pri             0           Time R_RDY high priority
tim_txcrd_z             0           Time TX Credit Zero (2.5Us ticks)
tim_txcrd_z_vc  0- 3:   0           0           0           0
tim_txcrd_z_vc  4- 7:   0           0           0           0
tim_txcrd_z_vc  8-11:   0           0           0           0
tim_txcrd_z_vc 12-15:   0           0           0           0
er_enc_in               0           Encoding errors inside of frames
er_crc                  0           Frames with CRC errors
er_trunc                0           Frames shorter than minimum
er_toolong              0           Frames longer than maximum
er_bad_eof              0           Frames with bad end-of-frame
er_enc_out              0           Encoding error outside of frames
er_bad_os               0           Invalid ordered set
er_rx_c3_timeout        0           Class 3 receive frames
discarded due to timeout
er_tx_c3_timeout        0           Class 3 transmit frames
discarded due to timeout
er_c3_dest_unreach      0           Class 3 frames discarded
due to destination unreachable
er_other_discard        0           Other discards
er_type1_miss           0           frames with FTB type 1 miss
er_type2_miss           0           frames with FTB type 2 miss
er_type6_miss           0           frames with FTB type 6 miss
er_zone_miss            0           frames with hard zoning miss
er_lun_zone_miss        0           frames with LUN zoning miss
er_crc_good_eof         0           Crc error with good eof
er_inv_arb              0           Invalid ARB
open                    0           loop_open
```

```
transfer                0          loop_transfer
opened                  0          FL_Port opened
starve_stop             0          tenancies stopped due to starvation
fl_tenancy              0          number of times FL has the tenancy
nl_tenancy              0          number of times NL has the tenancy
```

Chapter Essentials

Host Bus Adapters and Converged Network Adapters Hosts interface with the FC SAN via either HBAs or CNAs. These PCI devices appear to the host operating system as SCSI adapters, and any storage volumes presented to the OS via them appear as locally attached SCSI devices. Both types of card also offer hardware offloads for FCP operations, whereas CNA cards also offer hardware offloads of other protocols such as iSCSI and TCP/IP.

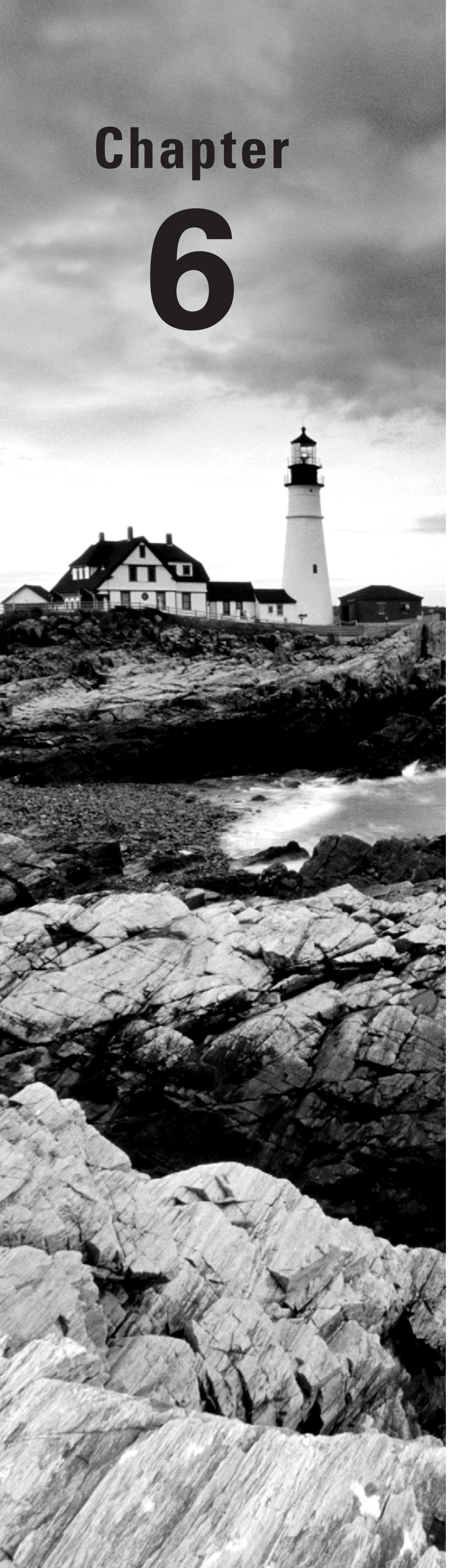
Switches and Directors FC switches and directors provide the connectivity between hosts and storage. Using switches for connectivity offers huge scalability as well as good performance and improved manageability. Switches provide fabric services to help partition the SAN and make it more manageable.

Inter-Switch Links Inter-switch links (ISLs) connect multiple switches together, allowing them to merge into a common fabric that can be managed from any switch in the fabric. ISLs can also be bonded into logical ISLs that provide the aggregate bandwidth of each component ISL as well as providing load balancing and high-availability features.

Virtual Fabrics Each SAN can be partitioned into smaller virtual fabrics, sometimes called VSANs. VSANs are similar to VLANs in the networking world and allow you to partition physical kit into multiple smaller logical SANs/fabrics. It is possible to route traffic between virtual fabrics by using vendor-specific technologies.

Summary

In this chapter we talked in depth about fibre channel storage area networks including why we have them and why they are still useful in today's world. We discussed the physical components of a SAN, including host bus adapters, cables, switches, and storage arrays. We also talked about many of the logical components in a SAN, such as fabrics, the name server, and zoning, and how everything—the physical and logical components—work together to form a resilient and high-performance fabric suitable for mission-critical storage. We also talked about the various SAN topologies, and their pros and cons, and detailed discussions on naming routing and flow control. Then we finished the chapter off with a discussion of some best practices and troubleshooting tips and examples.



iSCSI SAN

TOPICS COVERED IN THIS CHAPTER:

- ✓ iSCSI from 40,000 feet
- ✓ Initiators and targets
- ✓ iSCSI interfaces
- ✓ IP network considerations
- ✓ Discovery
- ✓ Login
- ✓ Sessions
- ✓ iSNS
- ✓ Security
- ✓ iSCSI gateways

In this chapter, you'll explore the increasingly popular iSCSI SAN. This chapter first covers the major concepts and principles and then discusses some technologies that will help you build and manage iSCSI SANs of varying sizes and complexities. And as iSCSI SANS are built on top of IP networks, this chapter covers the important network configuration options that will help you build, maintain, and perform a highly available and secure iSCSI SAN. As there are a lot of network options, I help map these to your individual requirements.

Finally, this chapter points out some of the mistakes that people have made when deploying iSCSI SANs and runs through some examples of configuring iSCSI in various operating systems.

iSCSI from 40,000 Feet

iSCSI is a storage networking technology that allows storage resources to be shared over an IP network. More often than not, the storage resources being shared on an iSCSI SAN are disk resources. However, as iSCSI is a mapping of the SCSI protocol over TCP/IP—exactly the same way that SCSI has been mapped over other *transports* such as Fibre Channel—it is theoretically possible for any SCSI devices to be shared over an iSCSI SAN. In reality, though, the vast majority of devices shared on an iSCSI SAN are disk devices or tape-related devices such as tape drives and tape changers.

NOTE The iSCSI SAN is a close cousin of the Fibre Channel (FC) SAN, sharing much of its DNA. This is good news if you are already familiar with FC SANs. But don't worry if you're new to all of this. By the end of this chapter, you will be able to more than hold your own.

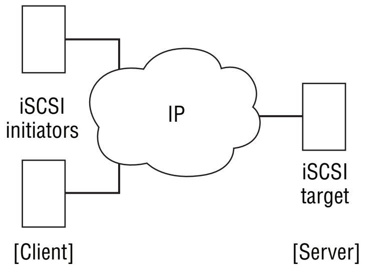
First up, let's get some of the basics out of the way.

iSCSI is an acronym for *Internet Small Computer System Interface*, and is correctly written with a lowercase *i* followed by uppercase *SCSI*, as follows: iSCSI. It is pronounced *eye-SKUZ-ee*. An iSCSI SAN deals with block storage and maps SCSI over TCP/IP. Although any SCSI device can theoretically be shared over an iSCSI SAN, it is almost always used for sharing primary storage such as disk drives. Backup devices are also quite common.

I always like a diagram, so let's take a high-level look at a simple iSCSI SAN. Figure 6.1 introduces the three major components in all iSCSI SANs:

- Initiators (servers)
- Targets (disk arrays or servers sharing storage over iSCSI)
- IP network

FIGURE 6.1 A simple iSCSI SAN



In an iSCSI SAN, initiators issue read data/write data requests to targets over an IP network. Targets respond to initiators over the same IP network. All iSCSI communications follow this request-response mechanism, and all requests and responses are passed over the IP network as iSCSI protocol data units (PDUs).

Although we will get into a lot more detail later in the chapter, it is important to note that the iSCSI PDU is the fundamental unit of communication in an iSCSI SAN. All iSCSI communication is via PDUs:

- Device discovery
- Device login
- Authentication
- SCSI commands
- SCSI data
- State-change notifications

iSCSI Is Cheap

People will often tell you that iSCSI is cheap. This can be both true and untrue, so you need to be careful.

When referring to iSCSI as cheap, people are usually comparing it to an FC SAN. While it is true that iSCSI *can* be deployed at a fraction of the cost of an FC SAN, the important word here is *can*. The reality is that an iSCSI SAN can be as cheap or expensive as you want it to be. But the adage that "you get what you pay for" holds very true for iSCSI SANs. High performance, high availability, and good security come at a cost. You could go for the mega-cheap option and throw your software initiators and software targets straight onto an existing shared IP network, cross your fingers, and hope for the best. Or you could deploy dedicated iSCSI HBAs in your servers and connect them to high-performance iSCSI storage arrays over a dedicated IP network with dedicated switches and IPsec for security. One of these options will give you high performance, high availability, and solid security. The other will not. The point is that an iSCSI SAN can be very cheap or very expensive, and what you choose *must* be driven by your requirements.

Now that you understand the basics, let's get under the hood.

Initiators and Targets

iSCSI SANs are composed of initiators and targets connected by an IP network. While initiators and targets are technically processes that run on the respective devices, most people refer to the server as the *initiator* and the iSCSI storage device as the *target*. In fact, rather than use these technical terms, people will be sloppy and use the terms *iSCSI host* and *iSCSI array* instead.

An iSCSI *target* is usually either a server with lots of disks attached or a dedicated *storage array*. In larger environments, the target tends to be a dedicated storage array, as these tend to provide the best performance, availability, and feature set. They also usually cost a lot more money than a standard server with a lot of shared disk.

iSCSI Interfaces

iSCSI initiators and targets require a physical interface to the network. These interfaces are usually PCI devices that are either integrated to the server motherboard or included as PCI expansion cards. They connect to the intermediate TCP/IP network via copper or fiber-optic cables.

When choosing an iSCSI interface, four major options are available. And as with most things in life, you tend to get more if you pay more. The question is this: Is what you get worth the extra you paid? The options, listed in order from cheapest to most expensive, are as follows:

Option 1: Standard Ethernet NIC with a Software Initiator This is the cheapest but probably most popular option. It takes a vanilla Ethernet NIC and converts it to an iSCSI initiator using host-based software called a *software initiator*. Most modern operating systems, including Linux and Windows, come with a built-in iSCSI software initiator, usually implemented as a kernel-based device driver. This makes configuring a server as an iSCSI initiator or target really cheap and really simple. However, they don't support the OS boot partition being on an iSCSI SAN, and they consume host CPU cycles, though the latter is becoming less and less of a concern with modern CPUs.

Option 2: Ethernet NIC with a TCP Offload Engine and a Software Initiator This is similar to option 1, with the major difference being that the Ethernet NIC has a TCP offload engine (TOE). This offloads processing of the TCP stack from the host CPU to the NIC, and on a busy NIC this can make a difference. However, iSCSI processing, such as building PDUs and encapsulation/decapsulation, is still handled by the host CPU. This is rarely seen in the real world.

NOTE Neither option 1 nor option 2 allows for diskless servers to boot from an iSCSI SAN without additional diskless boot software such as CCBoot.

Option 3: iSCSI Host Bus Adapter This takes things to the next level. This option offloads all the TCP and iSCSI processing from the host CPU to the processor on the host bus adapter (HBA). This increases the cost but removes the reliance on the host CPU to perform iSCSI functions. Also, from a feature perspective, iSCSI HBAs usually come with an option-ROM that allows diskless servers to be booted from the iSCSI SAN.

Option 4: Converged Network Adapter Using a converged network adapter (CNA) is similar to option 3. It offers everything that the iSCSI HBA offers (increased cost, reduced impact on host CPU, and boot from SAN) but has the added versatility of being dynamically configurable for protocols other than iSCSI, such as FCoE.

Option 1 is the most common in the real world and is extremely simple to set up in home and other small lab environments. All of these options appear to the OS as SCSI adapters. Therefore, the OS (aside from the software initiator, if any) is unable to distinguish volumes presented via the iSCSI SAN from volumes on a local disk inside the server. This makes using iSCSI SANs with existing operating systems simple and easy.

Deciding which of these is best for your environment may not be so simple. This will depend entirely upon your requirements. For example, standard vanilla NICs with software initiators and targets over a nonisolated IP network may work well in labs and small or remote offices. However, you wouldn't run your mission-critical line-of-business applications on that kind of infrastructure. For that, you want to be looking at something more along the lines of iSCSI HBAs and dedicated IP networks.

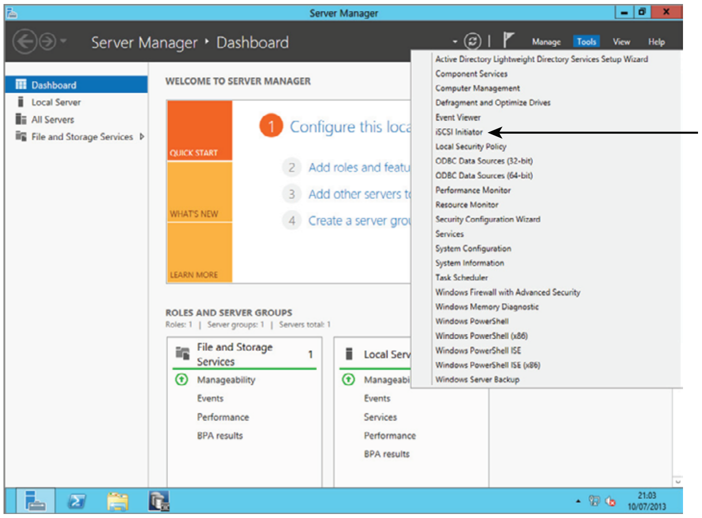
Exercise 6.1 shows how you configure the Windows Server 2012 software iSCSI initiator.

EXERCISE 6.1

Configuring the Windows Server 2012 Software iSCSI Initiator

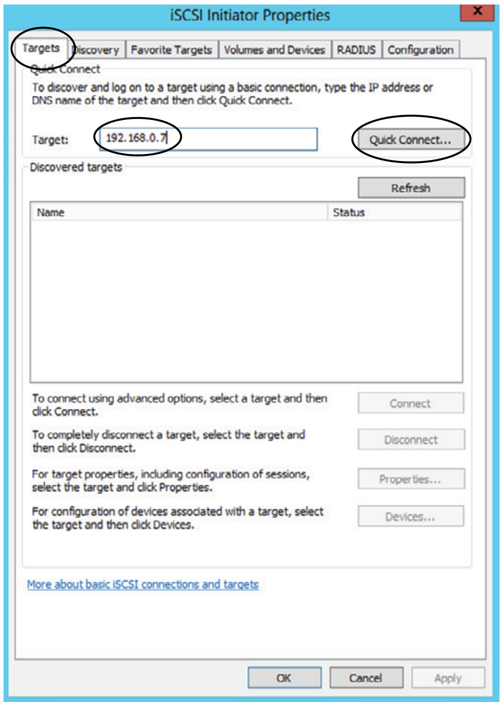
In this example, you'll walk through the process of enabling the software iSCSI initiator included in the Windows Server 2012 operating system. At this stage in the chapter, assume a simple unauthenticated connection to an iSCSI target. You will learn how to configure authentication later in the chapter.

1. From within the Windows Server Manager interface, select the Tools option from the top right-hand corner, as shown in the following image.



2. If the iSCSI service is not already running, you will be prompted to start it. Click Yes if prompted.

3. On the resulting screen, on the Targets tab, type the IP address of the DNS name of your iSCSI target array in the Target field. Then click Quick Connect, as shown in the following image.



To discover and log on to a target using a basic connection, type the IP address or DNS name of the target and then click Quick Connect.

Target: 192.168.0.7

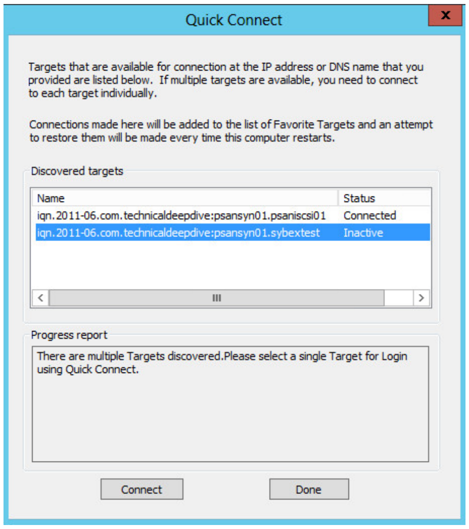
To connect using advanced options, select a target and then click Connect.

To completely disconnect a target, select the target and then click Disconnect.

For target properties, including configuration of sessions, select the target and click Properties.

For configuration of devices associated with a target, select the target and then click Devices.

4. The Quick Connect option returns a list of iSCSI targets running on that IP address or DNS name. From that list, select the target you wish to connect to. In the following image, the `iqn.2011-06.com.technicaldeepdive.psansyn01.sybextest` target is selected. Once you have selected your desired target, click Connect.



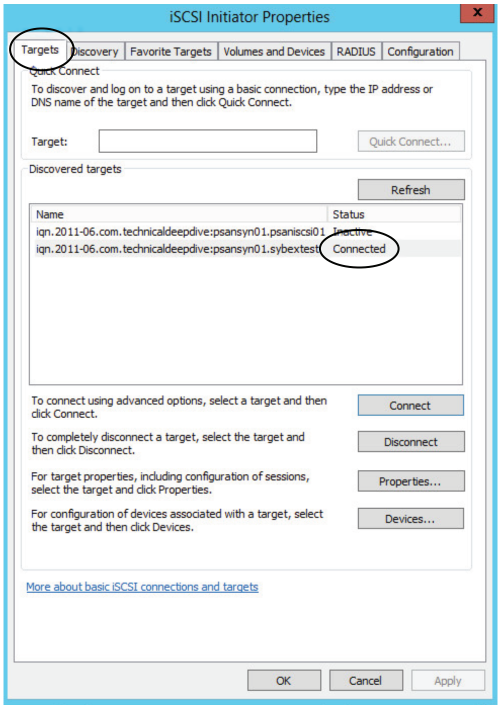
Targets that are available for connection at the IP address or DNS name that you provided are listed below. If multiple targets are available, you need to connect to each target individually.

Connections made here will be added to the list of Favorite Targets and an attempt to restore them will be made every time this computer restarts.

Name	Status
iqn.2011-06.com.technicaldeepdive:psansyn01.psaniscsi01	Connected
iqn.2011-06.com.technicaldeepdive:psansyn01.sybextest	Inactive

There are multiple Targets discovered.Please select a single Target for Login using Quick Connect.

5. Click OK on the following screens, as at this point you don't want to configure any advanced features.
6. The iSCSI initiator now tries to connect to the selected iSCSI target. If successful, it will return you to the iSCSI Initiator Properties screen on the Targets tab and show the selected iSCSI target as Connected, as shown in the following image.



iSCSI Initiator Properties

Targets | Discovery | Favorite Targets | Volumes and Devices | RADIUS | Configuration

Quick Connect

To discover and log on to a target using a basic connection, type the IP address or DNS name of the target and then click Quick Connect.

Target: Quick Connect...

Discovered targets

Refresh

Name	Status
iqn.2011-06.com.technicaldeepdive:psansyn01.psaniscsi01	Inactive
iqn.2011-06.com.technicaldeepdive:psansyn01.sybextest	Connected

To connect using advanced options, select a target and then click Connect. Connect

To completely disconnect a target, select the target and then click Disconnect. Disconnect

For target properties, including configuration of sessions, select the target and click Properties. Properties...

For configuration of devices associated with a target, select the target and then click Devices. Devices...

More about basic iSCSI connections and targets

OK Cancel Apply

You have now configured the Windows software iSCSI initiator and connected it to an iSCSI target. You can now use iSCSI LUNs available on that target.

IP Network Considerations

The IP network is crucial to an iSCSI SAN, as it is absolutely fundamental to performance, security, and availability. If you get this part of your iSCSI environment wrong, you are in for a world of hurt!

NOTE

TCP/IP is a great protocol that provides in-order delivery and error recovery, works over just about every medium known to man, and is extremely widely deployed and well-known. This can make it a great choice as a transport for SCSI. But it is vital that you understand the significance of its role in an iSCSI SAN.

Network Options

Plenty of networking options are available. The options presented next are in order from cheapest to most expensive. Again, as with iSCSI interfaces, you get what you pay for. The following list is not exhaustive, and other network options are available, but these are the most obvious and most common:

Option 1: Shared Vanilla IP Network This is the cheapest, simplest, least secure and least performant option. In this configuration, iSCSI traffic shares the same L2 broadcast domain with all other network traffic. It works and it's simple, but it is usually suited for only lab and home-office environments. If the IP network is unstable or very busy, expect performance problems and complaining users and colleagues. As an example, users backing up their iTunes library over the network in the middle of the business day can cause major packet loss and bring iSCSI performance to its knees.

Option 2: Dedicated VLAN This option, with a dedicated network connection for the iSCSI traffic on each server, offers segregation of iSCSI traffic from other network traffic. This can increase performance, and if the VLAN is nonroutable, it offers modest levels of security. As this option usually doesn't require any additional networking kit, it can keep costs down while providing decent uplifts in performance and security.

Option 3: Dedicated Physical Network This is the safe pair of hands. It offers the best in security, performance, and availability and is the most common choice in the real world. But as you may have guessed, it comes in at the highest cost. As with the iSCSI HBA or CNA, if you are deploying a mission-critical line-of-business app on an iSCSI SAN and want to be able to sleep at night, this is the option for you. An added layer of safety would be to have two dedicated switches with separate paths, effectively creating a network A and a network B, as we do in the FC world. This option would be so safe that you could sleep well at night and then also during the day!

These three aren't all of the options, but they constitute the basis of the most widely used available networking options.

Here is some key advice about general network configuration: First, go for option 2 or option 3—preferably option 3. It will save you a whole load of trouble. Next, keep the network configuration as simple as possible; don't try to be clever. Routing for inter-site connections is fine, but don't route your iSCSI network to your user network. For many vendors, it is also a best practice to disable Spanning Tree Protocol (STP) on ports connecting to iSCSI initiators and targets. Each vendor will have their own best practices around STP, but it is common for it to be disabled on ports connecting to iSCSI initiators and targets. If your network requires some form of STP, Rapid Spanning Tree Protocol (RSTP) may be preferable with PortFast enabled on all ports connecting to iSCSI initiators and targets so that these ports will immediately be in STP-forwarding mode when network events occur. BPDU Guard can also be used with ports that have PortFast enabled so that devices connected to them are prevented from participating in STP.

You may have heard that jumbo frames—Ethernet frames larger than the default maximum of 1,514 bytes, up to as much as 9,000 bytes—are a must for iSCSI. The implementation of jumbo frames can (and often does) cause problems, especially if all switches and devices in your network aren't configured with the same jumbo frame configuration. If you're confident at networking or have a good networking team, you may want to do this, as it can bring performance benefits. However, if you're not certain what you're doing in this area, you may find it's not worth the trouble.

iSCSI over a WAN or LAN

While it is possible for the IP network to be a corporate WAN or even the Internet, the vast majority of deployments are to a corporate LAN. This is so much the case that iSCSI may have been more appropriately named *Intranet SCSI* rather than Internet SCSI. This favoritism of the LAN comes from both a performance as well as a security perspective. Remember, storage traffic and storage traffic requirements are vastly different from most traditional uses of TCP/IP networks.

Real World Scenario

Overlooking Network Requirements

One of the most common mistakes in the real world, notably among those with little iSCSI experience, is overlooking network requirements.

A company deployed iSCSI straight onto the existing shared corporate network, based on the assumption that all that is needed is *any old IP network*. This resulted in unpredictable and often poor performance. Weeks were wasted investigating the root cause of the performance problem, and once it was identified as an under-specced network, it involved additional cost for a dedicated network kit after the project budget was closed.

iSCSI PDU

Since we've been talking networking, let's take a quick look at the iSCSI PDU and how data is encapsulated at the various layers involved in iSCSI.

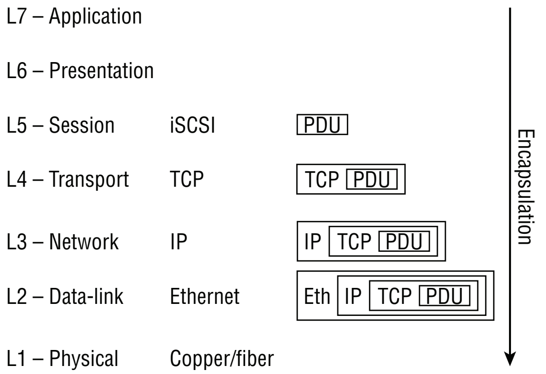
Looking a little further under the hood, the iSCSI PDU is made up of some mandatory and some optional segments. The most important of these segments is the *basic header segment (BHS)*, which is present in every iSCSI PDU. It is a fixed 48-byte field that contains the SCSI *command descriptor block (CDB)* and associated target LUN info. It is always present and is always the first segment. Quite often it is the only segment in a PDU, as all other segments are optional, including PDU CRC/digests, and are usually omitted. Generally speaking, you will never have to know more than this about the makeup of an iSCSI PDU. Integrity is ensured via TCP checksums and Ethernet CRCs.

Encapsulation

iSCSI operates at the Session layer of the Open Systems Interconnect (OSI) reference model.

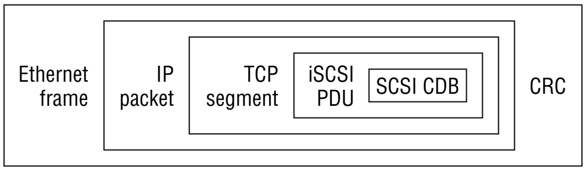
The life of an iSCSI I/O operation is pretty much as follows: An application issues an I/O operation to the OS/filesystem, which sends it to SCSI. SCSI creates the CDB and passes this to the iSCSI layer, which incidentally SCSI sees as just any old transport. CDBs and associated target LUN data are encapsulated by iSCSI into PDUs at the Session layer and passed further down the layers until they hit the wire for transport across the network. As they are passed down the different layers, additional layers of encapsulation are added. Figure 6.2 shows how iSCSI PDUs are encapsulated within TCP segments and then IP packets and then finally Ethernet frames.

FIGURE 6.2 iSCSI PDU encapsulation



There is nothing iSCSI-specific about the encapsulation once below the Session layer; TCP headers are used for reliable delivery, and IP headers for packet routing. Figure 6.3 shows an encapsulated iSCSI PDU.

FIGURE 6.3 Encapsulated iSCSI PDU



There is no implicit 1:1 mapping of iSCSI PDUs to TCP segments. An iSCSI PDU can span more than one TCP segment, IP packet, and Ethernet frame. Also, because of the nature of IP networks, frames, and therefore PDUs, may arrive at the target out of order. This is not a problem, as the TCP stack on the receiving end takes care of reordering segments and ensuring in-order delivery to the iSCSI layer.

NOTE Within an iSCSI session, command sequencing is tracked via *command sequence numbers (CmdSN)*. The CmdSN is incremented one digit for each command in a sequence.

Once received by the target, each of these encapsulation layers is stripped off, including the iSCSI PDU encapsulation, until all that is left is the SCSI CDB. This encapsulation/decapsulation will occur fastest on systems that have iSCSI HBAs or CNAs, as the encapsulation/decapsulation is performed by the HBA or CNA and not by the host CPU.

If iSCSI PDU size exceeds maximum segment size of the TCP layer, the PDU will be fragmented across multiple TCP segments. This *can* be fixed by manipulating the Maximum Transmission Unit MTU size on your network, but be careful. Do this only if you know what you are doing, as it breaks the cardinal rule of "keep things simple."

NOTE Of course, the underlying transport for the IP network could be any link-layer protocol and does not have to be Ethernet. However, it almost always is, so throughout this chapter, we assume it is Ethernet.

iSCSI Names

Every device on an iSCSI SAN needs a name. No name, no identity on the iSCSI SAN. iSCSI names are tied to iSCSI nodes and not a specific interface card in a node. This is important, as it allows the name to remain with the node even if NICs and HBAs are swapped around between devices.

iSCSI names are also permanent and globally unique. They do not change if the IP address or fully qualified domain name (FQDN) of a node or card changes.

iSCSI Qualified Names

Despite there being three iSCSI naming conventions available, the *iSCSI Qualified Name (IQN)* is by far and away the most popular and widely deployed. With this in mind, this section concentrates on the IQN and then touches on the other two types.

IQNs can be up to 233 bytes in length, are not case sensitive, cannot contain spaces, and are globally unique. They are also location independent, meaning that if an iSCSI node's IP address or hostname changes, the IQN is not affected. In fact, an iSCSI device can keep its IQN for its entire life. An IQN, however, is not burned in like an Ethernet MAC address or FC World Wide Name. It can be easily changed by an administrator if required.

iSCSI names are hierarchical, and within an IQN, three characters have special meaning:

- The dot `.`
- The colon `:`
- The dash `-`

These characters partition the iSCSI name. You will see how they are used later in this section.

Uniqueness of IQNs is achieved via two means. First, the IQN includes a domain name that must be owned by the company owning the iSCSI SAN. Second, to avoid situations where a company may have bought the rights to a domain name previously used by another company, a date is included in the IQN. This should be a date from the time the company deployed the iSCSI SAN-owned the rights to the domain name. While it is possible to ignore these guidelines, it is highly recommended that you follow this practice if changing iSCSI names!

NOTE More often than not, companies don't bother changing the default IQN that their devices come configured with. And this is one good way of ensuring uniqueness, as the vendor will have already followed the guidelines outlined in this section. For example, an EMC VNX array at a customer site has the following default IQN on one of its iSCSI ports: `iqn.1992-04.com.emc:cx:ckm00124XXXXXXX.a10`.

NOTE Interestingly, domain names used in iSCSI names do not have to be resolvable on the Internet or on the local corporate LAN. DNS resolution plays no part in defining and using iSCSI names. The reason that the domain name must be owned by the company to whom the iSCSI SAN belongs is purely to maintain global uniqueness of iSCSI names.

Now that you have some of the basics, let's take a look at an example IQN:

```
iqn.2011-06.com.technicaldeepdive:server:lablinuxdb01
```

While this might look a little complicated at first, it absolutely isn't.

IQNs names are broken into four fields:

- Type
- Date
- Naming authority
- Customizable string

Looking at our example, the type is `iqn`, identifying this name as an IQN type name (we will discuss the other two types shortly).

The date is generally the year and month at which the company registered the domain name that will be used in the name. The date format must be *YYYY-MM*. It is this combination of registered domain name and date at which the domain name was registered that ensures global uniqueness.

The naming authority is the domain name in reverse format. *Reverse format* means that the extension comes first, in this case `com`, followed by the `.` divider, followed by `technicaldeepdive`.

The customizable string can be any UTF-8 text that you as an administrator decide. However, as when choosing server names, it is highly recommended that you give some thought to defining a meaningful and scalable naming convention.

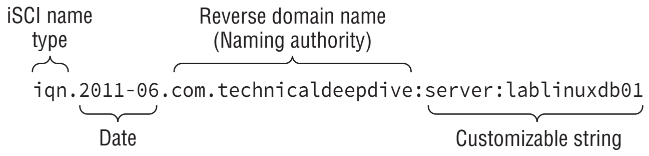
The format is as follows:

```
type.YYYY-MM.reversedomainname:customizablestring
```

Type is always followed by a period (`.`), the year is always separated from the month by a dash (`-`), the date and the reverse domains name are always separated by a period (`.`), and the reverse domain name and the customizable string are always separated by a colon (`:`). What comes after the colon in the customizable string is pretty open. You are then free to use the special characters if you wish, but from this point on in the name, they no longer have any special meaning.

Now that you know the major components and delineators that make up an IQN, let's take another look at our previous example as presented in Figure 6.4.

FIGURE 6.4 IQN


iSCI name type
Reverse domain name (Naming authority)
iqn.2011-06.com.technicaldeepdive:server:lablinuxdb01
Date
Customizable string

Alternative iSCSI Naming Conventions

Alternatives to the IQN naming standard include *Extended Unique Identifier (EUI)* and *T11 Network Address Authority (NAA)*. Both provide global uniqueness but are far less popular than IQN, and neither are human readable, at least not by normal human beings.

The EUI name format (EUI-64) consists of the letters `eui`, followed by a period (`.`), followed by 16 hex digits. For example:

```
eui.0123456789ABCDEF
```

The value of the 16 ASCII hex digits is formed by combining the Organizationally Unique Identifier (OUI) assigned to the company by the Institute of Electrical and Electronics Engineers (IEEE), and an extension identifier assigned by the company. The two combined result in a 64-bit (16 hex character) unique ID. The most significant bits represent the OUI, and the least significant bits the extension ID. It is unlikely that you will have the misfortune of having to deal with EUI names.

The NAA naming format was added to iSCSI in 2005 because both FC and SAS supported NAA naming formats and iSCSI did not. The thinking behind adding NAA was interoperability with FC and SAS. It is mercifully rare in deployment. However, just in case you come across this in an exam, the NAA naming format is as follows: `naa`, followed by a period (`.`), followed by what is referred to as the *NAA value*. The NAA value can be either 64-bit or 128-bit. 64-bit NAA values are 16 hex characters, whereas 128-bit NAA values are 32 hex characters. NAA values are not human readable.

iSCSI Aliases

Aliases can also be assigned to an initiator or target, allowing you to assign an even more user-friendly name to devices. However, these cannot be used in place of proper iSCSI names for operations such as discovery, login, and authentication. They are useful only for assigning more human-readable names to iSCSI devices in management tools and for similar purposes.

Wrapping up naming, IQN format is by far the most widely deployed and by far the most human friendly. EUI and NAA formats are horrific, but mercifully rare in their deployment.

Device Discovery

For an initiator and target to communicate, the initiator needs to be able to discover the target. To do this, the initiator needs to know three things about the target: IP address, TCP port, and iSCSI name. This information is obtained via the *discovery* process.

There are a few methods of discovery, intended to suit different-sized iSCSI SANs. Here are the various options, starting with those suited to the smallest environments and finishing with those suited most to large environments:

Manual Device Discovery In the very smallest environments, you can manually configure initiators with a file that lists available targets (IP address, TCP port, and iSCSI name). But this is neither dynamic nor scalable, and for these reasons it is rarely used. Technically speaking, no discovery is actually performed, as the list of targets is preconfigured on the initiators.

SendTargets This assumes prior knowledge of the IP address and TCP port used by the network portal of the target. The initiator issues the `SendTargets` command to the network portal on the target, and the target responds (request-response) with a list of available targets. This is common in small to medium iSCSI SANs and configurations that employ iSCSI gateways, which are rare and are discussed at the end of the chapter.

Automatic Device Discovery The final and most scalable options are those where the initiators have no prior knowledge of any targets on the SAN and must seek out targets on their own. Service Location Protocol (SLP) is the first and requires an SLP agent to be running on the initiator and the target. The SLP user agent on the initiator issues multicast messages to locate targets, and the service agent on the target responds. SLP is ideal for small to medium iSCSI SANs but not for large SANs. To be honest, it's not deployed that often. For large SANs, iSNS is recommended. iSNS is discussed in more detail in the "iSNS" section later in this chapter.

Once an initiator has discovered a target, the next step is to perform an iSCSI login with that target.

iSCSI Sessions

Once iSCSI devices (initiators and targets) are connected to the IP network and initiators have discovered targets, they can start the process of establishing sessions. A *session* is a communication path between a single iSCSI initiator and a single iSCSI target. Sessions contain at least one, and potentially more, TCP connections. It is over these TCP connections that commands and data are sent, in the form of iSCSI PDUs. This might sound like a lot to take in, but it's simple and by the end of this chapter it will be crystal clear. Let's take a look in more detail.

NOTE

To be technically accurate, when discussing iSCSI communications, iSCSI transfer direction should be described from the viewpoint of the initiator, meaning that outbound iSCSI traffic is heading away from the initiator, and inbound traffic is heading toward the initiator. Granted, this is a minor point, but worth knowing.

Device Login

Before iSCSI sessions can be created, and therefore before data can be read and written, an initiator and a target need to perform the *iSCSI login* process. iSCSI login is a two-phase process:

1. Security negotiation
2. Parameter negotiation

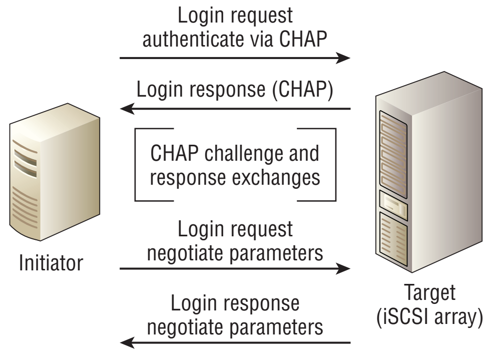
Of these two phases, security negotiation is optional. However, if security negotiation is to happen, it must happen before the negotiation of other session parameters.

The login process is started by initiators issuing iSCSI login request PDUs to the IP address and TCP port of the iSCSI target. This port can be the default well-known IANA-assigned port 3260/tcp, or a user-configured port. In iSCSI parlance, this combination of an IP address and listening TCP port on an iSCSI target is known as a *network portal*.

On receipt of iSCSI login requests, the target responds with iSCSI login responses. As we should expect by now, login requests and login responses are formed as iSCSI PDUs, and follow the familiar request-response model common throughout iSCSI.

Figure 6.5 shows the entire login, authentication, and parameter negotiation process.

FIGURE 6.5 High-level iSCSI login phases



Authentication

While it is not mandatory, it is *highly recommended* that initiators and targets authenticate each other. After all, IP networks are notoriously prone to attacks such as spoofing and man-in-the-middle attacks—although deploying dedicated isolated networks goes some way to avoiding man-in-the-middle attacks. To get around this, Challenge-Handshake Authentication Protocol (CHAP) should be used for authentication.

Although CHAP can help with authenticating initiators and targets, it does nothing to protect data in flight; data is still sent in cleartext. Therefore, if you have serious concerns about security, you should consider other network-level precautions, such as isolated networks and encryption technologies such as IPsec. However, while IPsec is great at providing cryptographic services for in-flight data, it can have a significant performance overhead.

NOTE While there is a feeling in the industry that CHAP is a relatively weak authentication method, using a random or otherwise strong password (shared secret) can significantly help. If you choose a simple password that is prone to dictionary attacks, the benefit of using CHAP is significantly reduced. Also, the iSCSI standards require that traffic flowing from the initiator to the target have a different shared secret than traffic flowing from the target to the initiator. This also helps avoid some of the more well-known weaknesses with CHAP.

Although other non-CHAP-based authentication options may sometimes be available, these should be considered with care. Because CHAP is the only security authentication mechanism specified in the standards, it is by far the most widely implemented and supported. The niche nature of other potentially stronger options makes them more difficult to implement and then support, as well as being subject to potential interoperability issues.

The best advice, if you are security conscious, is to deploy to an isolated nonroutable network, use CHAP for authentication with strong shared secrets, and use IPsec to encrypt the data in transit. It is also common practice to keep the management interfaces on a separate management network/VLAN. The point is that there are many options, and you can pick and choose which are appropriate for your requirements.

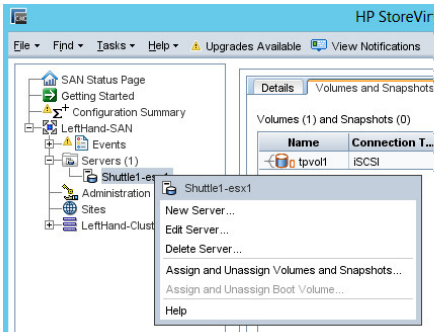
Let's now run through configuring a CHAP authentication on an iSCSI target (array) and an iSCSI initiator in Exercise 6.2 and Exercise 6.3.

EXERCISE 6.2

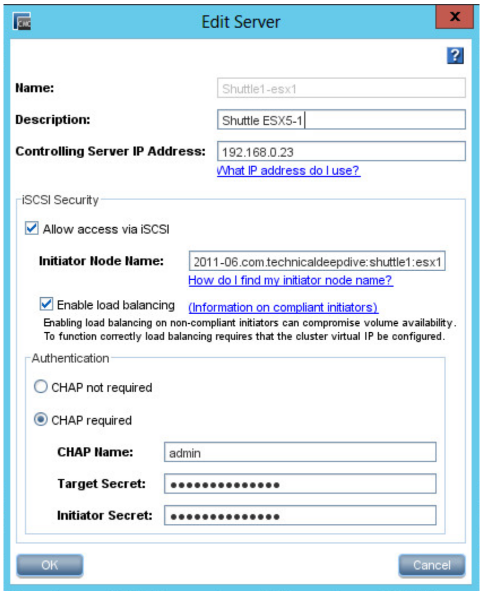
Configuring CHAP Authentication on an iSCSI Target (Array)

In this exercise, you will configure a target on an HP StoreVirtual iSCSI array to require CHAP and mutual CHAP authentication.

1. In the GUI for our StoreVirtual iSCSI array, select the iSCSI target from the tree pane. Right-click the target (server) and choose Edit Server from the context menu, as shown here.



2. In the Edit Server configuration screen, select the CHAP Required radio button in the Authentication area toward the bottom of the screen. Enter a username and a password for the Target Secret and Initiator Secret. Make sure you remember these passwords.



Name: Shuttle1-esx1
Description: Shuttle ESX5-1
Controlling Server IP Address: 192.168.0.23
What IP address do I use?

iSCSI Security
[✓] Allow access via iSCSI
Initiator Node Name: 2011-06.com.technicaldeepdive:shuttle1:esx1
How do I find my initiator node name?
[✓] Enable load balancing (Information on compliant initiators)
Enabling load balancing on non-compliant initiators can compromise volume availability.
To function correctly load balancing requires that the cluster virtual IP be configured.

Authentication
CHAP not required [✓ CHAP required]
CHAP Name: admin
Target Secret: ••••••••••••••
Initiator Secret: ••••••••••••••

You have now configured CHAP authentication for the `Shuttle1-esx1` iSCSI target on the HP StoreVirtual iSCSI array. This target will no longer accept unauthenticated requests to connect/log in.

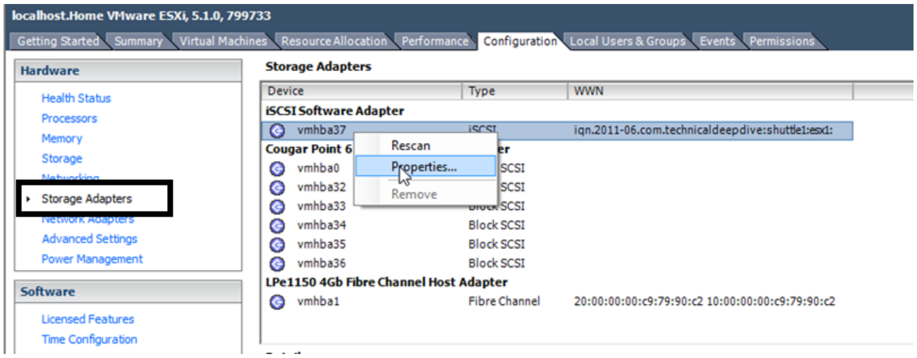
You can download evaluation copies of the HP StoreVirtual VSA as a VMware ODF file to play around with in your home lab.

EXERCISE 6.3

Configuring CHAP Authentication on VMware ESXi iSCSI Initiator

In this exercise, you will walk through the steps of configuring CHAP authentication on VMware ESXi running the software iSCSI initiator.

1. Log in to vCenter, or directly to the ESXi host, navigate to the Storage Adapters section of the Configuration tab to access the properties.


localhost.Home VMware ESXi, 5.1.0, 799733

Getting Started | Summary | Virtual Machines | Resource Allocation | Performance | Configuration | Local Users & Groups | Events | Permissions

Hardware

- Health Status
- Processors
- Memory
- Storage
- Networking
- Storage Adapters
- Network Adapters
- Advanced Settings
- Power Management

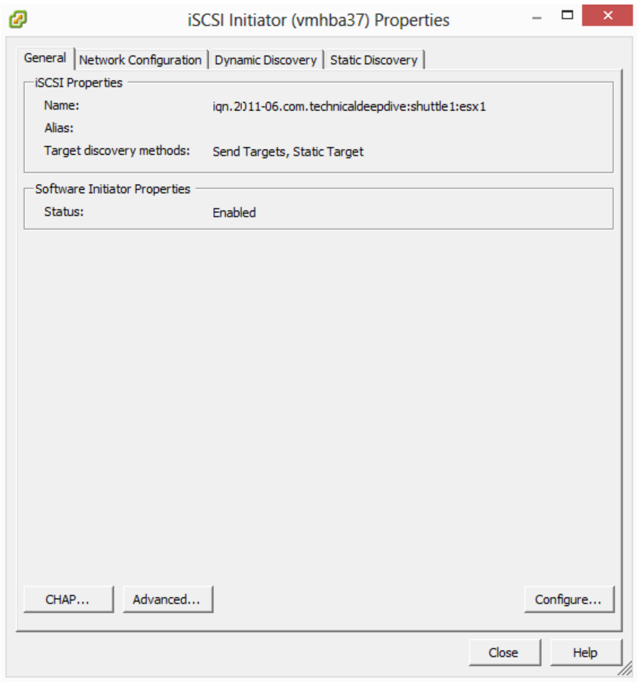
Software

- Licensed Features
- Time Configuration

Storage Adapters

Device	Type	WWN
iSCSI Software Adapter		
vmhba37	iSCSI	iqn.2011-06.com.technicaldeepdive:shuttle1:esx1:
Cougar Point 6		
vmhba0	SCSI	
vmhba32	SCSI	
vmhba33	Block SCSI	
vmhba34	Block SCSI	
vmhba35	Block SCSI	
vmhba36	Block SCSI	
LPe1150 4Gb Fibre Channel Host Adapter		
vmhba1	Fibre Channel	20:00:00:00:c9:79:90:c2 10:00:00:00:c9:79:90:c2

2. On the resulting Properties screen, click the CHAP button in the bottom-left corner.


iSCSI Initiator (vmhba37) Properties

General | Network Configuration | Dynamic Discovery | Static Discovery

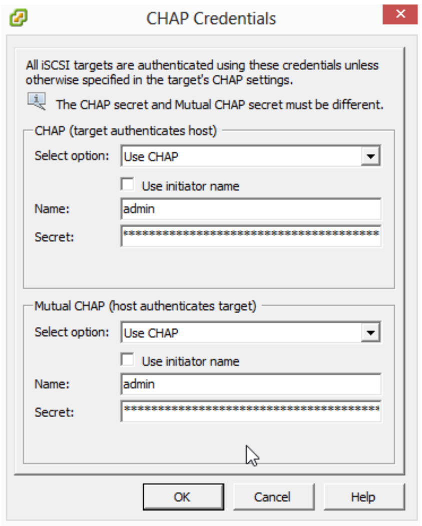
iSCSI Properties

Name:	iqn.2011-06.com.technicaldeepdive:shuttle1:esx1
Alias:	
Target discovery methods:	Send Targets, Static Target

Software Initiator Properties

Status:	Enabled

3. The CHAP Credentials screen appears. In the CHAP (Target Authenticates Host) area, choose the Use CHAP option from the Select Option drop-down menu. In the Mutual CHAP (Host Authenticates Target) area, make the same selection. Enter the Name and Secret credentials. Then click OK.



CHAP Credentials

All iSCSI targets are authenticated using these credentials unless otherwise specified in the target's CHAP settings.

The CHAP secret and Mutual CHAP secret must be different.

CHAP (target authenticates host)

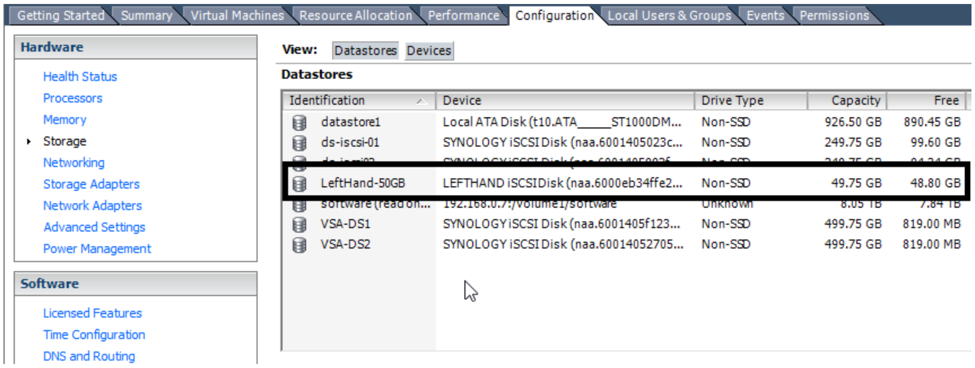
Select option:	Use CHAP
	☐ Use initiator name
Name:	admin
Secret:	**

Mutual CHAP (host authenticates target)

Select option:	Use CHAP
	☐ Use initiator name
Name:	admin
Secret:	**

4. A rescan of the HBA will be required. Click Yes on this screen.
5. Once the rescan has completed, go to the Storage area of the Configuration tab in your vCenter GUI, and you will see your iSCSI LUNs.

In the image, you have a new 50 GB iSCSI LUN visible from the HP StoreVirtual iSCSI array. If you look closely, you will see that the device is showing as a LEFTHAND iSCSI-Disk. This is because HP has rebranded their LeftHand product as StoreVirtual, but under the covers the device still identifies itself as LeftHand.



Datastores

Identification	Device	Drive Type	Capacity	Free
datastore1	Local ATA Disk (t10.ATA_____ST1000DM...	Non-SSD	926.50 GB	890.45 GB
ds-iscsi-01	SYNOLOGY iSCSI Disk (naa.6001405023c...	Non-SSD	249.75 GB	99.60 GB
ds-iscsi02	SYNOLOGY iSCSI Disk (naa.6001405003f...	Non-SSD	249.75 GB	94.24 GB
LeftHand-50GB	LEFTHAND iSCSIDisk (naa.6000eb34ffe2...	Non-SSD	49.75 GB	48.80 GB
software (read on...	192.168.0.7:/volume1/software	Unknown	8.05 TB	7.84 TB
VSA-DS1	SYNOLOGY iSCSI Disk (naa.6001405f123...	Non-SSD	499.75 GB	819.00 MB
VSA-DS2	SYNOLOGY iSCSI Disk (naa.60014052705...	Non-SSD	499.75 GB	819.00 MB

Once the iSCSI login process is complete, and assuming it is successful, the session enters what is known technically as *full-feature phase (FFP)*.

Full-Feature Phase

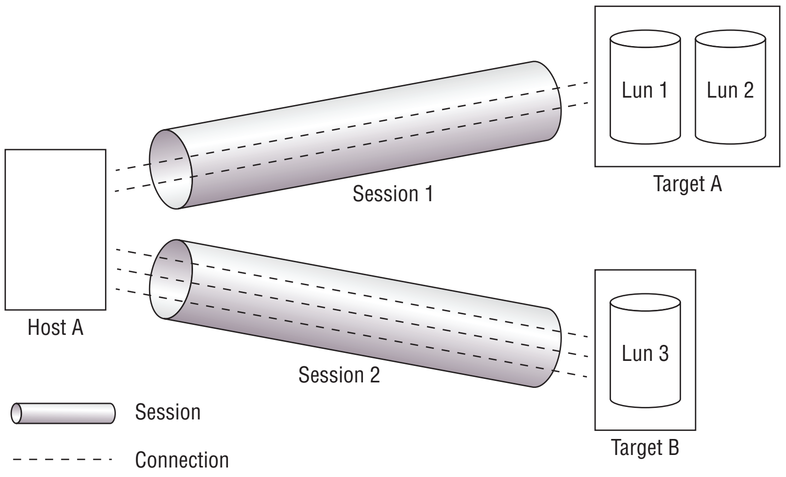
Full-feature phase is where the action happens. Once a session is in this phase, commands and data can be sent between initiators and targets. Prior to this, only login-related PDUs are permitted between the initiator and target.

It is time for a quick recap of the iSCSI login process. iSCSI login is the process that negotiates security, builds the TCP connections, and defines all the parameters that will be used when the session enters FFP. The login process is composed of a set of request-response PDUs between an initiator and a target. First up, the TCP connections are established, and then the iSCSI login session. After successful login, the session enters FFP, and from then on initiators and targets are free to go about their everyday business of reading and writing data.

FFP is composed of one or more iSCSI sessions. These sessions represent communication between an initiator and a target. Each session is composed of one or more TCP connections, and only a single initiator and a single target can participate in a single session. If an initiator wants to communicate with more than one target—let's say, for instance, two targets—it must have at least two sessions (one session per target).

Let's take a look at a quick example of an initiator communicating with two targets. Host A is a database server with two LUNs from Target A, which is a high-performance iSCSI storage array. LUN1 is used for the database, and LUN2 is used for logs. Host A also has a single LUN presented from Target B, which is a deduplicating storage array. This LUN (LUN 3) is used for a backup area. It is perfectly normal for Host A to have connections to these two targets, but in order to do so, there will need to be a minimum of two sessions—one for the communications between Host A and Target A, and the other between Host A and Target B. See Figure 6.6. Each of these sessions may have one or more TCP connections.

FIGURE 6.6 iSCSI sessions



Sessions are identified by a unique session ID (SSID) comprising an initiator component known as the initiator session ID (ISID) and a target component known as the target portal group tag (TPGT). Each TCP connection is uniquely identified within the context of a single iSCSI session by a connection ID (CID). This CID is used during login and logout and various other operations.

Within a session, related requests and responses must have what is known as *connection allegiance*. Connection allegiance requires that all related requests and responses occur over the same TCP connection. For example, if an initiator makes a READ request over TCP Connection X, the response from the target to the initiator must be over TCP Connection X. The same goes for WRITE requests issued from an initiator. And just in case it hasn't sunk in yet, all communication within an iSCSI session is done via iSCSI PDUs. It is the iSCSI layer (Session layer of the OSI model) that builds PDUs and manages all sessions.

Initiators are responsible for performing logouts. In the real world of an iSCSI administrator, it is not necessary to know intimate details of this process.

NOTE In the real world, many popular iSCSI arrays, such as Dell EqualLogic, present a separate target for each LUN, simplifying LUN masking.

iSNS

iSNS stands for *Internet Storage Name Service*, and if you are administering a large data center with many iSCSI arrays, you may want to deploy iSNS. And if you do so, you will want to know it well. It is a godsend to large iSCSI environments.

At a high level, iSNS is a simple centralized database containing configuration data about your iSCSI SAN. Initiators and targets can register information with the iSNS as well as query information from it. Querying the iSNS allows devices on the iSCSI SAN to determine the topology of the iSCSI SAN as well as capabilities and properties of other devices that are visible to it. Initiators and targets can also register attributes with the iSNS, making it a dynamic repository that can greatly simplify iSCSI SAN management. Initiators register with the iSNS and query it for a list of available targets. Targets register with the iSNS so that they can be discovered by initiators. Both can also register for state updates from the iSNS.

NOTE If you know FC SAN and are familiar with the FC Simple Name Service (SNS), you are 99 percent of the way to understanding iSNS, as iSNS is modeled on FC SNS.

iSNS provides the following services in an iSCSI SAN:

- Name service
- Partitioning of the iSCSI SAN into discovery domains
- Login services
- State-change notifications
- Mapping FC to iSCSI devices

Two of these merit further discussion—partitioning and state-change notifications. Before we do this, let's take a quick moment to cover some of the basic components of the iSNS service.

The protocol used by iSCSI devices to communicate with the iSNS is *iSNSP (iSNS Protocol)*. It is a lightweight request-response protocol, and the default TCP port that the iSNS listens on is 3205. All communication between initiators, targets, switches, and the iSNS occurs via iSNSP. This is, of course, subsequently encapsulated within the obligatory TCP segments, IP packets, and Ethernet frames for delivery across the IP network.

The iSNS client running on each initiator or target is the process responsible for locating the iSNS service as well as all communication with the iSNS.

The iSNS server hosts the iSNS database and is responsible for making that database available to the iSCSI SAN, so that devices can register with and query information from it.

The iSNS database is the information store for the configuration of the iSCSI SAN. This is a simple lightweight database and does not require complex and expensive stand-alone database servers. It can be Lightweight Directory Access Protocol (LDAP) integrated. It can be dynamically updated by initiators and targets.

Now that you know the major components that make up and interact with the iSNS, let's take a closer look at the services iSNS provides.

Discovery Domains

Discovery domains are a way of partitioning your iSCSI SAN into smaller, more manageable, and more secure zones. Each of these zones is referred to as a discovery domain, or DD for short.

In a nutshell, initiators and targets are administratively assigned to discovery domains. Once assigned to a discovery domain, devices can only discover, log in, and communicate with other devices that share a common discovery domain. It is possible, and common, for devices to be members of multiple DDs, but the main point to note is that in order for an initiator and a target to communicate with each other, they must be members of a common DD.

Partitioning your iSCSI SAN into smaller discovery domains has several advantages. The first advantage is security. When DDs are implemented, devices can see and communicate only with other devices in the same DD. Not only can they not log in to other devices outside their discovery domains, they will not even know of the existence of devices outside their discovery domains. This leads to a second important advantage of discovery domains. By restricting the view of the SAN, they restrict the number of devices that can be discovered and logged in to. As an example, let's assume a wide-open iSCSI SAN (an administrator's nightmare!) with no discovery domains implemented. In this wide-open example, if no discovery domains were implemented, you may get situations where every initiator on the iSCSI SAN attempts to log in to and query every other device on the iSCSI SAN. This would be time-consuming and wasteful of resources, not to mention a security risk.

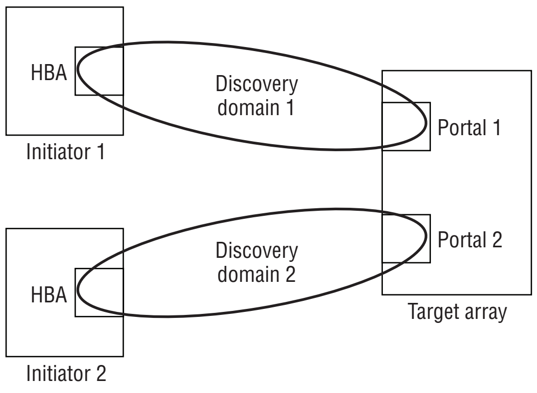
NOTE The implementation of discovery domains relies on the initiator playing by the rules, and the target rejecting logins that come from devices not in the appropriate discovery domain. Discovery domains on their own do not have a mechanism to strictly enforce them. For example, there is no mechanism to drop network packets from an initiator to a target that it does not share a discovery domain with. In effect, there is nothing inherent in a discovery domain to physically stop a misbehaving initiator from breaking the rules. In this sense, discovery domains are a form of security by obscurity. Initiators can't mess around with targets they don't know about.

As a security precaution, it is strongly recommended that you operate a closed-door policy when it comes to discovery domains. This is the opposite of a wide-open policy. In a closed-door policy iSCSI SAN, new devices on the iSCSI SAN are automatically registered with no discovery domain, leaving them effectively off the iSCSI SAN until you explicitly configure them into a discovery domain. This is a best practice in both iSCSI and FC SANs.

NOTE From a security perspective, iSNS and discovery domains restrict only device discovery. Masking of devices on a target, CHAP for authentication, and IPsec for privacy of in-flight data are all still necessary for the ultimate in security. Discovery domains absolutely do not do away with the need for these other technologies.

Figure 6.7 shows a small iSCSI SAN with two initiators, Init1 and Init2, and a single target with two network portals. The iSCSI SAN is divided into two discovery domains. In this example, Init1 is unaware of the existence of Init2.

FIGURE 6.7 Discovery domains



As an administrator, you can place one or more *network portals* of a given network entity into a discovery domain. Think of network portals as one or more interfaces on an iSCSI target. By placing network portals into discovery domains, you are controlling which interfaces (network portals) will send and receive traffic to and from particular initiators. Looking back at Figure 6.7, we can see that network portal 1 (P1) will permit traffic to and from Init1 but not Init2. Likewise, network portal 2 (P2) will permit traffic to and from only Init2 and not Init1.

NOTE A *portal group* is a collection of network portals within a single network entity/iSCSI target. This is not too dissimilar to a NIC team. It is possible for sessions to have multiple connections that span multiple portals within a portal group. A *target portal group tag* is just a numerical string used to identify a target portal group.

A quirk worth noting is that if you place an entire network entity into a DD rather than explicitly placing a particular network portal into the DD, all portals for that network entity will permit traffic for that discovery domain.

For a discovery domain to be active and effective (in use), it must be a member of the active discovery domain set (DDS). A DDS contains many discovery domains, and the discovery domains that are members of the active DDS are the only active and operational discovery domains.

State-Change Notifications

State-change notifications (SCNs) are a method of relaying relevant configuration and topology information to devices on the iSCSI SAN.

The SCN service runs on the *iSNS server*, meaning that you need to be running an iSCSI SAN with iSNS to benefit from SCNs. For day-to-day configuration and administration of your iSCSI SAN, that is probably about as much as you will need to know. However, there is more to it that may be of benefit when troubleshooting or in exams.

The iSNS issues SCNs in response to events and changes on the iSCSI SAN. It is the *iSNS client* running on the initiator or target that is responsible for registering with the iSNS for SCN updates. If an initiator or target does not register itself for SCNs, it will not receive them.

Only two types of SCN exist:

- **Regular SCNs** are issued to all registered devices within a discovery domain. So, in order for a device to receive a regular SCN, it must have registered to receive them, and reside in a discovery domain to which the SCN is issued. This twofold approach keeps SCN traffic to a minimum and ensures that the messages that do reach a device are relevant.
- **Management SCNs** can be registered for by only known *control stations*, meaning that regular initiators and targets cannot register for management SCNs. Also, the distribution of management SCNs is not restricted to the boundaries of discovery domains. As an administrator, you can control the list of authorized control stations (also known as *management stations*).

Registration for SCNs is done at iSCSI login time.

Administering the iSNS

A *management station*, sometimes referred to as a *control station*, is a special entity on an iSCSI SAN with carte-blanche access to the iSNS. This administrative access to the iSNS is usually out-of-band (OOB), meaning that it is accessed over an interface (IP address) that is separated from the real iSCSI traffic and usually dedicated just to management traffic. Some of the common administrative functions performed by a management station include the following:

- Creating, enforcing, and destroying discovery domains
- Maintaining the list of authorized control stations
- Maintaining the Management SCN setting
- Setting whether the default discovery domain and DDS are enabled or disabled (recommend disabled!)

There is a dedicated iSNS Management Information Base (MIB), *iSNSMIB*, for managing and monitoring iSNS via Simple Network Management Protocol (SNMP).

As the iSNS can become so crucial to the daily operation of your iSCSI environment, it needs appropriate hardware resources. Overloading the iSNS can result in SCNs not being generated and other problems.

Security

As security is such a hot and vital topic, it's worth taking a few minutes to quickly recap some of the security-related principles and best practices that we've already discussed as well as mention one or two more.

The iSCSI standard outlines the use of CHAP as a mechanism for initiators and targets to authenticate each other. However, CHAP does not have the greatest reputation in the network security world. The reality is that if weak CHAP shared secrets (passwords) are used, then CHAP will deter only the very crudest of attacks. However, if strong secrets (not using common words that can be found in a dictionary, including numbers, and so on) are used, it can be a useful security precaution. Therefore, it is highly recommended to use CHAP and to use it with strong secrets. However, even with strong secrets, it is important to note that CHAP provides only initiator and target authentication—that is, ensuring that the initiator and target are who they say they are. It does absolutely nothing to secure data transmitted over the wire.

So even with a well-configured CHAP solution, iSCSI still transmits data over the network as cleartext with absolutely no native cryptographic services. So if you don't use something like IPsec, all your data will be transferred in plain text and easily readable by anyone listening in. If this is a concern, you should seriously consider IPsec. If it's not a concern, you probably don't need IPsec.

As an additional security measure, it might be good practice to operate a dedicated physically isolated network, or nonroutable VLAN, for iSCSI traffic. You should also keep management interfaces off the iSCSI network.

If using iSNS, it is recommended to implement discovery domains and disable the default discovery domain. Doing so will mean that only devices that are administratively added to discovery domains will be able to discover and communicate with other devices on the SAN.

Most iSCSI targets also support LUN masking, where resources (LUNs) on a target are masked and visible to only specific initiators. LUN masking should always be used.

Other security measures exist, including stronger authentication methods, but many are niche and can be hard to support when the going gets tough. Implementing the preceding recommendations will be enough to satisfy most security requirements and ensure you can sleep well at night knowing you don't have any gaping security holes in your iSCSI SAN.

iSCSI Gateways

iSCSI gateways are used when an existing Fibre Channel storage array, with only FC ports, exists in an environment and you want to connect iSCSI initiators to it. The iSCSI gateway acts the way a network gateway should, by performing protocol translation between iSCSI and Fibre Channel Protocol (FCP). On one side of the gateway are the Ethernet port for the iSCSI connectivity, and on the other side are FC ports for the FCP traffic. It is the gateway that is configured as the iSCSI target, and not the FC storage array.

This is sometimes called a *bridged topology* or a *gateway topology* and is not as common as native iSCSI environments. This kind of configuration adds components and complexity to an iSCSI and FC SAN and should not be favored over a *native* iSCSI SAN.

What is more common than an iSCSI gateway configuration is a storage array that has both iSCSI and FC connectivity (sometimes referred to as a *multiprotocol array*). These tend to be high-end, very reliable systems and do not suffer from the added complexity issues that gateway configurations do. A simple iSCSI gateway configuration is depicted in Figure 6.8.

Chapter Essentials

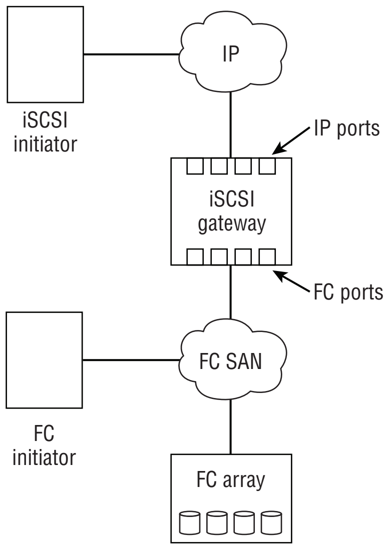
Understanding iSCSI Protocol Conceptually speaking, iSCSI is a SAN protocol that maps SCSI over IP and operates a simple request-response model where all messages between initiators and targets occur via iSCSI protocol data units, often referred to as simply PDUs or messages.

iSCSI Performance iSCSI performance is influenced by three main components. Best and most consistent initiator performance is achieved with dedicated iSCSI HBAs. Best and most consistent target performance is achieved by purpose-built iSCSI arrays. Best and most consistent network performance is achieved by dedicated network switches.

Encapsulation and IP SCSI commands are encapsulated at each layer of the network stack for eventual transmission over an IP network. The TCP layer takes care of transmission reliability and in-order delivery, whereas the IP layer provides routing across the network.

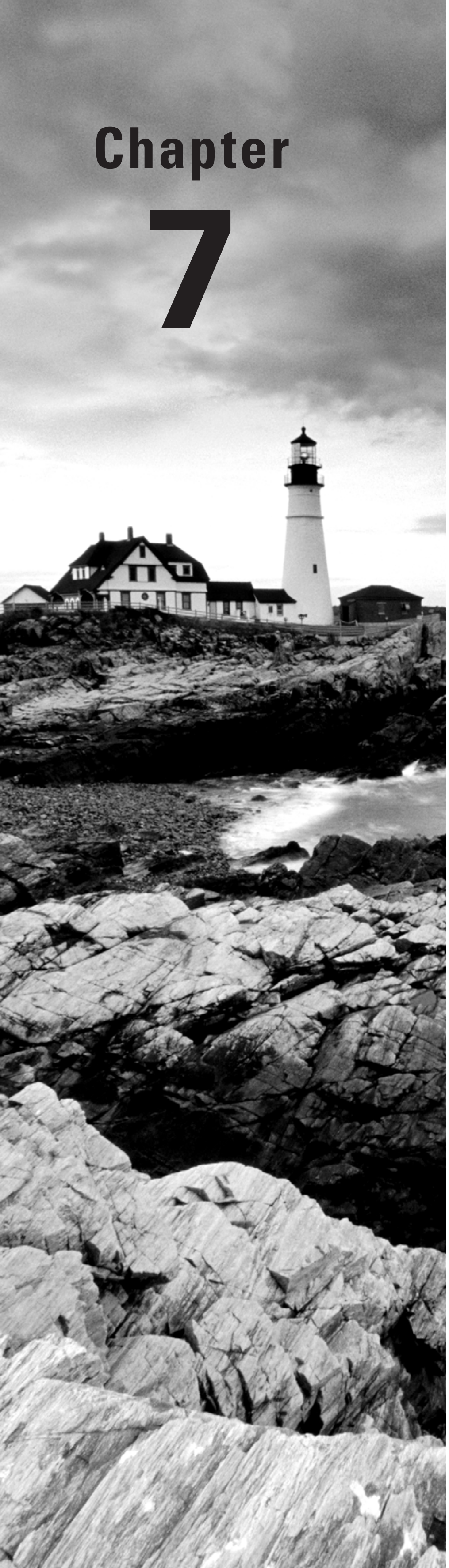
iSCSI and Security Multiple layers of security should be implemented on an iSCSI SAN. These include: CHAP for authentication, discovery domains to restrict device discovery, network isolation, and IPsec for encryption of in-flight data.

FIGURE 6.8 iSCSI gateway configuration



Summary

In this chapter you learned about the basics, as well as some of the advanced features, of iSCSI Storage Area Networks. We talked about iSCSI initiators and the different initiator options available, as well as different network options. When talking about initiators and networks, we discussed how each of the options influenced performance, availability, and cost. We also talked about some of the theory that is fundamental to iSCSI SANs, including iSCSI qualified names (IQN), device discovery, login, authentication, state change notifications, and discovery domains.



Files, NAS, and Objects

TOPICS COVERED IN THIS CHAPTER:

- ✓ NAS protocols including NFS and SMB/CIFS
- ✓ Traditional NAS arrays
- ✓ Unified storage arrays
- ✓ Scale-out NAS
- ✓ Object storage devices
- ✓ Content-aware storage
- ✓ Compliance archiving

In this chapter, you'll look closely at the major protocols involved with NAS, file-serving, and object storage environments. You'll start out by looking at traditional file-serving and NAS environments, before moving on to more-modern and far more interesting scale-out NAS technologies. You'll then move on to the modern and extremely interesting world of object storage.

This chapter covers NFS and SMB/CIFS file-serving protocols, including the differences between the two and some of the features available in the various versions of each. You will then compare and contrast the use of NAS appliances with traditional general-purpose file servers. Finally, you will examine object storage and object storage devices, including how they compare to traditional file services and how they play well with Big Data and cloud storage.

What Files, NAS, and Objects Are

In the storage world, there have traditionally been two types of storage:

- Block storage
- File storage

Block storage shares raw devices and can be thought of as working with bits and bytes, whereas *file storage* operates at a higher level, and as the name suggests, works with files and folders.

If you install a new disk drive in your PC, it will appear to the operating system as a *raw block device* that you will need to format with a filesystem before you can start using it. If your grandmother needed a second disk drive installed in her PC, you would probably have to do it for her. File storage is different and a lot simpler. If somebody shares a folder with you over the network, once you are connected to the network, the shared folder is ready to use. There is no need to mess around formatting it or doing anything technical like that.

Shared file storage is often referred to as *network-attached storage (NAS)* and uses protocols such as NFS and SMB/CIFS to share storage. Generally speaking, NAS storage is often used for unstructured data storage such as shared folders and document repositories. A notable exception is using a NAS array to provide NFS datastores for VMware.

SAN and NAS have been around for years. SAN shares storage at the block level, and NAS shares storage at the file level. Between them they rule the roost in most corporate data centers, and probably will for a lot of years to come. However, they are not best suited to the kind of scale demanded by the cloud and Web 2.0–type media such as the large image and video files that are exponentially growing in the social media space.

This is where a relatively new type of storage, known as *object storage,* comes into play. Object storage is the new kid on the block and is still finding its place in the world. So far, it has been a massive success in the public cloud arena, but less so in the traditional corporate data center (although it is seeing more and more use cases and demand in the corporate data center).

File/NAS storage and object storage are covered in greater detail throughout the chapter.

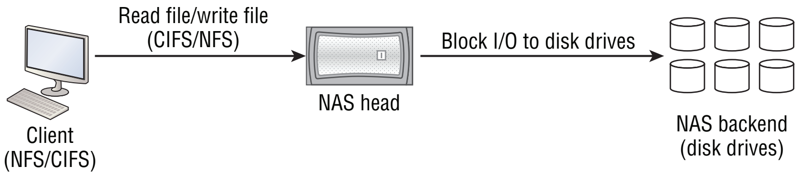
Network-Attached Storage

As established in the introduction to the chapter, NAS is not SAN. It might use the same three letters for its acronym, but the two are not the same! SAN works at the block level, whereas NAS works at the file level. As this chapter concentrates on file-level storage, we will talk only of files, and then later we will talk about objects.

As already stated, NAS is an acronym for *network-attached storage*. In the corporate data center, most people think of NetApp when they think of NAS. True, NetApp can do block and file storage, but it is mainly known for doing file storage. However, NetApp is not the only vendor of NAS storage technology out there.

Files are a higher level of abstraction than blocks. Although NAS clients talk to NAS filers in terms of files, the filer itself still has to communicate with the disk drives on its backend via block-based protocols. Figure 7.1 shows a simple NAS setup with the client mounting a filesystem exported from the front-end of the NAS appliance, while on the backend the NAS performs the required block storage I/O. Effectively, the NAS hides the block-level complexity from the clients.

FIGURE 7.1 NAS with hidden block access on the backend



The most common examples of NAS storage in the corporate data center are user home directories and team drives. In the Microsoft Windows world, these NAS drives are referred to as *shares*, *network shares,* or *CIFS shares*. In the UNIX world, they are usually referred to as *exports* or *NFS exports*. You will explore CIFS and NFS further shortly.

In the Microsoft Windows world, it is most common to *mount* a share to the root directory of a drive letter. For example, your user home directory might be mounted to `H:\`, and your department's team drive might be mounted to the root of `F:`. Windows also supports *mount points*, technically called *reparse points*, that allow external filesystems to be mapped to arbitrary locations within an existing filesystem—kind of like how it works in the Linux/UNIX world.

In the UNIX world, NFS exports are usually mounted to a mount point within the root (`/`) filesystem rather than a drive letter. For example, your team drive might be mounted to a directory called `tech` in the `/mnt` directory.

NOTE A *mount point* is a directory in a local filesystem, where a new external filesystem is made accessible. The directory that is used as the mount point should be kept empty. For example, your Linux server might have a `/teams` directory on its local hard drive but use this directory as a mount point for accessing an external filesystem that is accessed over the network.

NOTE In the Microsoft Windows world, the process of mounting a network drive is referred to as *mapping a drive.*

In both the Microsoft Windows and UNIX worlds, once a remote filesystem is mounted into the local namespace (either a drive letter in the Windows world or a mount point in the UNIX world), you read and write files to them as if the files and folders were local to your machine.

Although NAS might work at a higher level than SAN (block), it is by no means simple. NAS has to deal with the following concerns that SAN does not have to deal with:

- File-level locking
- User and group permissions
- Authentication
- Antivirus software

Now let's take a closer look at the most common NAS and file-sharing protocols.

NAS Protocols

Within the scope of NAS storage, there are several protocols. However, the most prominent and important—and the ones that you need to know the most about for the CompTIA Storage+ exam—are Network File System and Server Message Block.

NFS

It's only right that we start with *Network File System (NFS)*. When Sun Microsystems invented NFS, it pretty much invented network file sharing as we know it today.

NFS is a client-server protocol that operates natively over TCP/IP networks and is deeply rooted (pun intended) in UNIX. However, it is also implemented in Microsoft Windows environments, albeit far less frequently than in UNIX environments.

There are two major versions of NFS in use today:

- NFSv3
- NFSv4 (including 4.1 and 4.2)

At the time of writing this book, deployment of NFSv3 still far outstrips deployment of NFSv4, even though NFSv3 was standardized in the mid-1990s and NFSv4 in the early 2000s.

The major components in an NFS configuration are the NFS server and the NFS client. The NFS server *exports* specific directories to specific clients over the network. The NFS client *mounts* the NFS exports in order to read and write from them. NFS servers can be either general-purpose UNIX/Linux servers, or can be NAS devices running an NFS service.

NFSv3 Overview

NFSv3 is a stateless protocol that operates over UDP or TCP and is based on the Open Network Computing (ONC) Remote Procedure Call (RPC) protocol.

NFSv3 is pretty awful at security. Basically, the standards for NFSv3 stipulate security as optional, resulting in hardly anybody implementing any of the recommended security features. Therefore, some organizations, especially financial institutions, try to avoid NFSv3.

NFSv3 is also tricky to configure behind and connect to through a firewall. This is because it utilizes the *port mapper* service to determine which network ports to listen and connect on. Basically, when an NFS server starts up, it tells the port mapper service which TCP/IP port numbers it is listening on for particular RPC numbers. This means that NFS clients have to ask the port mapper service on the NFS server which TCP or UDP port number to contact the NFS server on for a given RPC number. This mapping can change if the NFS daemon is restarted. This can get a bit messy in firewalled environments.

Despite these deficiencies, NFSv3 is still a highly functional and popular file-serving protocol, so long as you can overlook the gaping security holes.

NFSv4 Overview

NFSv4 is a massive step forward for the NFS protocol, bringing it up to par with other network file-sharing protocols such as SMB/CIFS. However, despite being ratified as a standard in the early 2000s, not many vendors have implemented it and even fewer customers have deployed it. That aside, NFSv4 offers some good features and it is gaining in popularity. Some of the major improvements include the following:

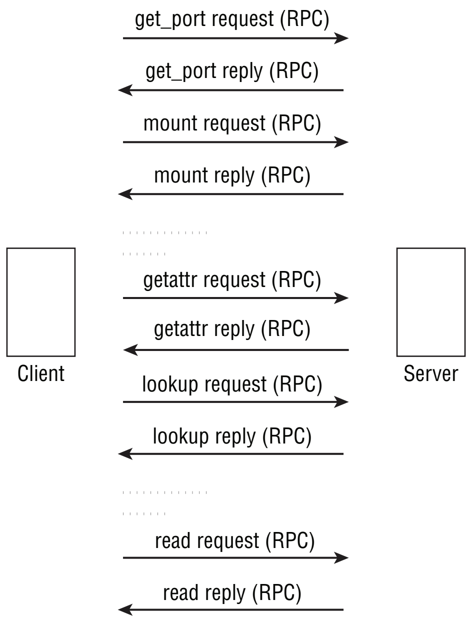
- Access control lists that are similar to Windows ACLs
- Mandated strong security
- Compound RPCs
- Delegations and client-side caching
- Operation over well-known TCP port 2049
- NFSv4.1 also brought parallel NFS (pNFS)

Let's look at some of these aspects in more detail.

NFSv4 Compound RPCs

NFSv3 was a chatty protocol—not the worst, but not great either. Basically, all actions in NFSv3 were short, sharp RPC exchanges through which the NFS client and NFS server would ping-pong lots of RPCs to each other. Figure 7.2 shows a typical conversation between a client and server using NFSv3.

FIGURE 7.2 Chatty NFSv3



It was typical for a simple mount of an export and subsequent read of a file on the exported filesystem to require 20 or more exchanges between the NFS client and server. That is not ideal on a LAN, and it is a nightmare over a WAN because of the high latency common in WANs. *Compound RPCs* in NSv4 fix this. The principle is simple. NFS clients batch operations and send them to the NFS server as fewer RPCs. Responses from the server get batched too. This results in a far less chatty protocol that is better suited for WANs as well as LANs.

Defining WAN, LAN, and MAN

A *wide area network (WAN)* is a network that covers a large geographic distance. It doesn't matter whether that network is privately owned or is linked over the Internet. For example, the London and New York offices of a company may be connected over a dedicated private circuit, while the New York and Cayman Islands offices may use a VPN tunnel over the Internet. Both can be termed WANs.

A *local area network (LAN)* is the opposite of a WAN. A LAN is usually restricted to a single building or very short distances (approximately 100 meters or so). In a LAN you can have high data rates such as 40 GB and higher and still have relatively low latency.

A *metropolitan area network (MAN)* fits somewhere in between a LAN and a WAN and generally operates over moderate distances such as a city/metropolitan area.

NFSv4 Security

NFSv4 does a much better job at security than NFSv3. Although NFSv3 *supports* strong security, people (including vendors) often failed to implement it. NFSv4 changes the game by *mandating* strong security.

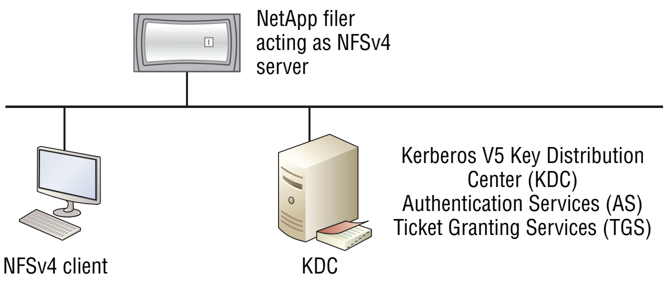
One of the major security enhancements is the use of Kerberos v5 for the usual cryptographic services:

- Authentication
- Integrity
- Privacy (encryption)

Both UNIX and Windows-based Kerberos Key Distribution Centers (KDCs) are supported.

Figure 7.3 shows an example NFSv4 configuration with a Kerberos KDC.

FIGURE 7.3 Simple NFSv4 configuration


NetApp filer acting as NFSv4 server
Kerberos V5 Key Distribution Center (KDC) Authentication Services (AS) Ticket Granting Services (TGS)
NFSv4 client
KDC

Mounting an NFSv4 export on a Linux host using Kerberos requires the mount options shown here:

```
mount -t nfs4 uber-netapp01:/vol/nfs4_test_vol /mnt/nfs4 -o sec=krb5
```

NFSv4 Access Control Lists and Permissions

With NFSv3, mount requests were allowed or denied based on the IP address of the client making the mount request. No thought was given to the user who was making the mount request. NFSv4 turns this on its head. In NFSv4, it is all about authenticating the user who makes the mount request.

NFSv4 has also caught up with SMB/CIFS with its use of proper access control lists (ACLs) that are based on the NTFS model. This is good because the NTFS security model is far superior to the traditional POSIX-based read/write/execute permissions common in the UNIX world.

ACLs in the NFSv4 world are populated with access control entries (ACEs). The model is a deny-unless-specifically-granted model, and because of this you should not have to explicitly deny access to files and directories. Abiding by this principle—not explicitly denying access—can keep troubleshooting ACL issues far simpler than if you do use explicit denies.

Putting It All Together

Now that you have explored the principles of NFS, let's go through an exercise of creating, exporting, and mounting an NFSv3 directory. See Exercise 7.1.

EXERCISE 7.1

Exporting and Mounting an NFS Share

On UNIX and Linux servers, the `/etc/exports` file contains the list of all NFS exported filesystems. If this file is present and configured, when a UNIX/Linux server starts up, the `nfsd` daemon will start and will export the filesystems listed. Naturally, I am using directory and client names that are on my system, and you will need to substitute names native to your system if you want to actively try this.

1. Logged on to your Linux server, `LegendaryLinux01`, use the `exportfs` command to export the `/home/npoulton/test_export` directory to a Linux client called `uberlinux01`:

```
# exportfs uberlinux01:/home/npoulton/test_export/
```

2. The previous command makes an entry in the `/etc/exportfs` file. Verify this with the `exportfs` command:

```
# exportfs
:/home/npoulton/test_export/uberlinux01
```

3. Now that your directory is exported to your Linux host called `uberlinux01`, let's switch to your `legendarylinux01` client and mount the export to a mount point of `/mnt/test`:

```
# mount legendarylinux01:/home/npoulton/test_export/ /mnt/test
```

4. Change the directory to `/mnt/test` and see whether you can see any files:

```
# cd /mnt/test
# ls -l
-rw-r--r--   1   npoulton is   11   May 9 10:58   uber_file
```

5. Unmount the export with the `umount` command:

```
# cd /
# umount /mnt/test
```

6. Switch back to your Linux export server (`legendarylinux01`) and unexport the directory. To do this, use the `-u` switch on with the `exportfs` command:

```
# exportfs -u uberlinux01:/home/npoulton/test_export/
```

WARNING

It is highly recommended to use root squashing when exporting resources via NFS. *Root squashing* sets the user ID of anyone accessing NFS exports from the user context of the root user on their local machine (the NFS client) to a value of the NFS server's `nfsnobody` account. This effectively ensures they don't get root privilege to the NFS export. Don't turn this behavior off unless you are absolutely certain you know what you are doing.

SMB/CIFS

First, let's get the lingo right. This is a tricky one, though. Technically speaking, the protocol is *SMB*, an acronym for *Server Message Block*. However, just about everybody refers to it as *Common Internet File System* (*CIFS*, pronounced *sifs*). It was originally called CIFS, but is now formally called SMB. Boring, I know! At the end of the day, few people care, but you're usually safe going with the flow, and *the flow* is calling it CIFS.

Also on the topic of terminology, you will often hear people refer to things like CIFS shares, CIFS clients, and CIFS servers. A *CIFS share* is any folder that is shared on the network using the SMB/CIFS protocol. This generally is anything you connect to using a UNC path such as this:

```
\\legendaryserver\sybex-book\
```

NOTE

UNC is an acronym for *Universal Naming Convention*, or *Uniform Naming Convention*. It is the syntax used to address CIFS shares over a network. The format for UNC paths is `\\`*`servername`*`\`*`sharename`*. Because UNC is used to address CIFS shares, it is predominant in Microsoft Windows environments.

CIFS is a client-server-based protocol. A *CIFS server* is any file server dishing out CIFS shares. This could be a Windows server that is sharing some of its local directories, or it could refer to an instance of a file server on a NAS array. A *CIFS client* is any device that accesses a CIFS share over the network. All Microsoft Windows machines have a CIFS client, and no additional software is required. Linux servers require a Samba client installed in order to access CIFS shares.

The Strengths of SMB/CIFS

CIFS is the de facto file-serving protocol in the Microsoft Windows world. It's been around for years and has been tuned and tweaked into what is now a best-of-breed file-serving protocol. CIFS operates natively over TCP/IP networks on TCP port 445.

CIFS is also widely deployed by many third-party NAS vendors, thanks to it being a publicly documented protocol despite the fact that it is Microsoft proprietary. Beware, though: not all third-party implementations implement all features.

CIFS is a pretty rich file-serving protocol. It supports all of the following, plus a lot more:

- Authentication
- Encryption
- Quotas
- Branch caching
- Advanced permissions
- Request compounding
- Request pipelining

NOTE

SMB/CIFS can be used to share and access devices other than files and folders. It can also share printers and be used for inter-process communication (IPC) between servers.

SMB/CIFS originally had a shocking reputation. It was extremely chatty and had a checkered history that didn't embrace TCP/IP. Then with the introduction of Windows 2000, Microsoft suddenly embraced TCP/IP as if it had invented it. However, although it was ugly in its early stages, it is much better looking now!

SMB 2.0 was a massive revision that really upped the ante. Major improvements included vastly reducing network-related chatter, reducing the overall number of commands in the protocol suite, introducing command compounding, and adding pipelining of commands.

Command compounding allows commands to be batched up and sent as fewer commands. The same goes for responses. *Pipelining* takes things a little further by allowing multiple compounded commands to be sent before an ACK is received. It also supports TCP window scaling. The net result is a superior, less chatty protocol that is more suited to WAN use cases, including the Internet.

WARNING

Be careful if using SMB/CIFS over the Internet, as there are many ways to exploit the protocol. You need to take care by making sure your servers and NAS devices are patched to the latest levels, and if possible you may want to utilize authentication and encryption using Kerberos.

SMB 3.0, introduced with Windows Server 2012, improved on the already excellent SMB 2.0 by adding various other improvements as well as introducing an interesting feature Microsoft calls *SMB Direct*. SMB Direct allows SMB traffic to be sent over high-performance, low-latency remote direct memory access (RDMA) technologies like InfiniBand.

On the security front, CIFS utilizes Kerberos for cryptographic services such as authentication and encryption, and it's pretty simple to set up. However, not all implementations of SMB/CIFS by NAS vendors implement Kerberos-based cryptographic services. Be sure to check before you buy.

All in all, modern versions of the SMB/CIFS protocol are solid. Microsoft is driving it hard, whereas many NFS shops—customers and vendors—have been very slow to implement and deploy many of the newer features that have been available in NFSv4 for over 10 years now.

A testament to the quality of recent versions of SMB/CIFS is that Microsoft now supports SQL Server 2012 database files to be stored on SMB/CIFS shares. This would not be supported if the protocol was not reliable and high performance.

Mapping a remote CIFS share is significant yet easy. See Exercise 7.2.

EXERCISE 7.2

Mounting a Remote CIFS Share

Here, you'll look at mapping and then checking the contents of a CIFS share. Obviously, if you follow these steps, you'll need to substitute the names of your own share and server.

1. Use the `net use` command to map a remote CIFS share on a server to a mount point using the UNC naming convention. Here, I am mapping TeamDrive on server W2012-01 to a mount point of `Z:\`, but you can use whatever remote CIFS share, server, and mount point you prefer.

```
net use Z: \\w2012-01\teamdrive
```

2. Now that the CIFS share is successfully mounted on the PC, run a `dir` command to list the contents of the share:

```
C:\>cd z:
Z:\>dir
 Volume in drive Z is 143.
 Volume Serial Number is D2B5-52F


 Directory of Z:\
21/09/2011  07:18    <DIR>          Users
```

As you can see from the `dir` output, the CIFS share contains a single directory called `Users`.

SMB/CIFS Permissions

SMB/CIFS file and folder security is based on access control lists that allow extremely granular control of file and folder permissions.

There are a few things to be aware of when working with permissions on CIFS shares. First, CIFS shares have two distinct sets of permissions:

- Share permissions
- NTFS permissions

Share permissions can be applied only at the root of the share, not to subfolders or individual files. Share permissions also apply only to connections over the network, meaning that anyone logged on locally (including Terminal Services) automatically bypasses any share permissions. NTFS permissions, on the other hand, can be applied to any file or folder, and apply equally over the network or for locally logged-on users. NTFS permissions are way more flexible and granular than share permissions, and most of the permissions work you do will be with NTFS permissions.

As every CIFS share has to have both types of permissions, how can they work effectively together? One very common approach is to configure share permissions to grant full control to the Everyone group, and then to start locking things down with NTFS permissions. This effectively bypasses share permissions.

There are a couple of important points that are common to both share and NTFS permissions:

- Both work on an implicit deny model. If access is not specifically granted, it will be denied.
- The most restrictive permission applies.

The second point basically means that deny permissions trump allow permissions. For example, consider a scenario where the user `npoulton` is a member of the IT-All security group and also a member of the IT-Contractors security group. If the IT-All group is allowed access to a particular folder but the ITContractors group is denied access, the security principal `npoulton` will not be allowed to access the folder.

However, it is not quite as simple as that. Explicit permissions trump inherited permissions, even inherited deny permissions. This means that if a folder inherits a deny permission from its parent folder for the IT-All group, but that same group is explicitly granted access to the folder, the explicitly granted access will override the inherited deny permission.

This works slightly differently when enumerating share and NTFS permissions. If share permissions allow user npoulton full control of a share, but then the NTFS permissions allow the same user only read-only access, the effective permissions for the npoulton user will be read-only access.

WARNING

Make sure you count to 10 and really think about what you are doing before granting NTFS full-control permission, as this permission allows the user to take ownership of the folder and its contents. Once a user has ownership of something, they are free to mess about with that object to their heart's content.

NOTE Clients and servers running different versions of the SMB protocol are said to be speaking different SMB dialects. When a client and server initiate communication, they negotiate the highest supported dialect.

FTP

FTP is an acronym for *File Transfer Protocol*. FTP is a client-server- based protocol that operates natively over TCP/IP networks. FTP is commonly used to transfer large files over the Internet, and as the name suggests, FTP actually transfers (moves) files between the client and the server. This differs from NFS and SMB, which allow files to remain on the server while they are being accessed and updated. Most web browsers support FTP, and there are also a wide variety of FTP clients available.

Vanilla FTP is weak on the security front and sends passwords as cleartext. If authentication and other security features are required, either FTP with SSL (FTPS) or Secure Shell FTP (SFTP) should be used.

In addition to the security weaknesses, FTP is a fairly primitive protocol when compared with the latest versions of NFS and SMB/CIFS. Consequently, FTP is losing popularity to protocols such as BitTorrent.

NAS Arrays

NAS arrays are specialized computer systems that act as file servers. They come in all shapes and sizes, starting with small home use systems and ranging all the way up to massively scalable, scale-out NAS systems. Most organizations run their businesses on mid- to high-end systems. These are often specialized, custom-built computers that pool and share large numbers of drives. Some NAS arrays have specialized custom hardware such as ASICs and FPGAs, although these are becoming less and less popular, and most are now built using commodity hardware. On the software side, things are different. Most NAS arrays run proprietary software that is stripped down to the bare bones and highly tuned specifically for file serving.

Running small, specialized NAS operating systems has several advantages, including these:

- High performance
- Small attack surface
- Simple upgrades and patch management
- Fewer bugs

High performance comes from the fact that the NAS OS does not have to cater to other general-purpose server requirements and can be tuned specifically for file serving. This also makes upgrading and patching significantly easier because you have fewer moving parts, so to speak. The small attack surface comes from the fact that the NAS OS runs only code specific to file serving and is therefore not at risk of hacks aimed at vulnerabilities in code that have nothing to do with file serving.

NOTE Application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) are effectively custom silicon—specialized processing hardware designed for a fairly specific task. There are a few minor differences between the two, but as far as we are concerned, they are *custom hardware* and are the antithesis to commodity hardware such as Intel x86 and x64 CPUs. There is a big move in the industry to get away from custom hardware and deploy as much as possible on commodity hardware, implementing the intelligence in software. This software-everything approach boasts lower costs and faster development cycles and is just about the only way to go for cloud-based technologies and software-defined technologies such as software-defined networking (SDN) and software-defined storage (SDS). There is still a small amount of custom hardware, ASICs, and FPGAs going into more-traditional storage technologies such as SAN and NAS disk arrays.

The primary goal of a NAS array is to pool storage and make it available to other systems as files and folders over an IP network. More often than not, the protocols used to share files are NFS and SMB/CIFS. Other protocols such as File Transfer Protocol (FTP) and Secure Copy (SCP) are sometimes included. Quite often, though, the protocol suites on NAS arrays lag behind the published standards and do not implement all features of a given protocol. For example, it is common for CIFS on a NAS device to lag behind the latest SMB/CIFS implementation on Microsoft Windows servers. This is due to several reasons. First, the NAS vendor will always lag behind Microsoft, as Microsoft owns and develops the protocol. The NAS vendors effectively copy its behavior. Second, not all NAS customers require all features, and vendors will implement only the features that their customers are asking for and the features for which they can justify the expense of development.

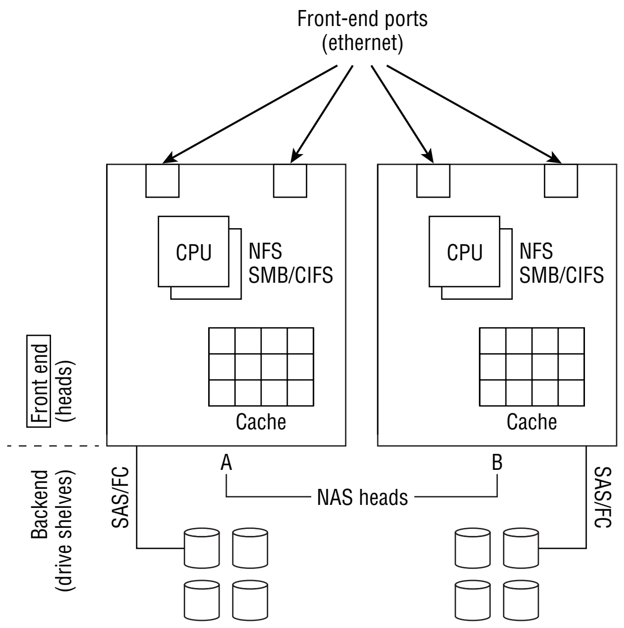
Although most NAS arrays are built on top of a commodity Intel architecture and often run a version of the Linux kernel, you cannot run your own software on them. The common exception is that some vendors allow you to run specialized NAS versions of antivirus software.

Figure 7.4 shows a logical block diagram of a typical NAS array.

The backend disk drives of a NAS array can be a part of the packaged NAS solution, or they can be LUNs from an external storage array. The latter is referred to as a *NAS gateway*, which is covered later in the chapter.

NOTE NAS arrays are also discussed throughout Chapter 3, "Storage Arrays."

FIGURE 7.4 NAS array block diagram



NAS Array vs. a General-Purpose File Server

NAS arrays are popular in modern data centers as consolidation points for file shares and storage for unstructured data.

NOTE

The term *structured data* refers to data that is highly organized in a known fashion, such as databases. On the other hand, *unstructured data* refers to most other types of data such as document repositories and rich media files like pictures and video. NAS can be used for both, but is more commonly used for unstructured data storage.

As mentioned earlier, a standard Windows or Linux server can be configured as a file server and perform the same functions as a NAS array. In fact, sometimes Windows and Linux servers support more features such as Microsoft branch caching and some of the advanced features of Microsoft Distributed File System (DFS). So why would you spend the money on a specialized NAS array?

NAS Pros

On the plus side, NAS arrays tend to provide benefits in four key areas:

- Simplified management
- Increased performance
- Increased data availability
- Advanced features

On the topic of simplifying management, NAS arrays usually have specialized GUIs and CLIs that make managing filesystems, network connections, and advanced features a lot simpler. It is common for multiple general-purpose file servers to be consolidated to a single NAS array, dramatically reducing the number of management points in an environment. Also, patching and code upgrades can be hugely simplified and usually done without taking the system offline, not to mention that hardware upgrades are usually simple and very expandable.

From a performance perspective, NAS arrays tend to pool lots of drives together on the backend. This leads to increased-performance and reduced-performance hot spots. NAS arrays can also come with large DRAM and flash caches to increase performance.

Most NAS arrays support multiple controllers—sometimes referred to as *heads*—that provide increased performance as well as data availability. These heads are often configured in an n+1 configuration, where a single head can fail but the NAS system can remain up and operating.

Some of the advanced features NAS arrays support include advanced RAID protection, snapshots, remote replication, deduplication, compression, and hypervisor integration.

NAS Cons

On the downside, NAS arrays do not always support all features of a given protocol. For example, not all NAS arrays support NFSv4. Also, not all NAS arrays support all features of SMB/CIFS. Commonly, features such as branch cache and certain aspects of DFS are often not supported.

Real World Scenario

Consolidating to a NAS array

A small/medium-sized business in the retail sector had an untidy sprawling file server estate built around an old version of Windows (Windows Server 2003) that was becoming more and more difficult to manage. The file server estate had started out using local disks in the physical file servers, but was starting to grow out of the locally attached disks and needed SAN storage presenting to the file servers in order to cope with growth in the file server estate. This SAN storage was not a cheap option, but there were no more slots for new disks in the file servers for more locally attached disk. It was decided to migrate the file server estate to NetApp Filers (NAS arrays). In the end, all eight of the company's physical file servers were migrated and consolidated to a single NetApp Filer. This Filer was a single point of management, had the capability to easily increase in capacity, and utilized replication and snapshot technologies to drastically improve the backup solution for the file server estate. Migrating file servers to the NetApp Filer was a huge success and the estate went from being out of control and sprawling, to under control with room for growth and major improvements to management, availability, and backup.

NAS vs. DAS

When comparing NAS to direct-attached storage (DAS), it is important to remember that DAS provides block storage, whereas NAS provides file storage—so it's not an apples-to-apples comparison.

Assuming you aren't concerned about the differences between block or file storage, the key benefits of NAS tend to be scalability and advanced features such as snapshots, remote replication, deduplication, and so on. However, NAS has higher latency and higher $/TB costs than DAS.

Sometimes NAS will be what you want, especially if you are looking for consolidated file serving, whereas for other requirements such as cheap dumb storage, DAS may be the better option.

NAS vs. SAN

This is more of an apples-to-apples comparison, although there is still the difference between file and block protocols.

Technically speaking, the differences boil down to the following. SAN storage shares only raw capacity over a dedicated network, whereas NAS also provides a network-aware filesystem (permissions, file locking, and so on). Each of those has pros and cons. SAN storage needs to be managed at the host by a volume manager layer and have a filesystem written to it, but shares from a NAS do not.

In the real world, NAS is sometimes seen as a bit of a dumping ground for office documents and rich media files. Although true, this does a huge disservice to NAS. It is probably fairer to say that NAS lends itself naturally to being shared storage for unstructured data, with structured data sets being more commonly deployed on SAN-based block storage.

Traditional NAS Arrays

Traditional NAS arrays like the original NetApp filers and EMC VNX devices are getting quite long in the tooth and are struggling to cope with many of today's unstructured data requirements. They tend not to be very scalable and can easily become a management headache in large environments, where it is not uncommon for medium-sized organizations to have 10 or more filers, and for larger organizations to have several hundred filers on your data center floor, each needing to be managed separately.

That aside, they do a good-enough job on a small scale and provide the usual NFS and SMB/CIFS protocols. They also provide features such as remote replication, snapshots, deduplication, and compression.

Unified NAS Arrays

Unified NAS arrays combine traditional NAS arrays with block SAN arrays in a single system. These hybrid arrays are sometimes called *multi-protocol arrays* because they support both file and block protocols. To some people, they are Frankenstorage, whereas for other people they are just what the doctor ordered. Generally speaking, they work for small and sometimes medium-sized organizations that have a requirement for block and file storage but cannot justify making two separate purchases. More often than not, these unified storage arrays maintain separate drive pools for block and file storage in order to keep the I/O of the two protocol types from walking all over each other and dragging performance down.

NOTE

Frankenstorage refers to storage technologies that appear to be crudely bolted together. Examples include unified storage arrays that have simply taken a block storage array and a filer head and bolted them together to look like a single unified system, whereas under the covers they are two separate systems.

Scale-Out NAS Arrays

Anyone who knows anything about traditional NAS arrays will tell you that they do not scale. If you have large NAS requirements, you are forced to deploy multiple discrete NAS arrays, each of which is individually managed. This results in what is commonly referred to as *NAS sprawl*.

Scale-out-NAS goes all-out to bring scalability to NAS. And it does this while riding the Big Data wave, where companies are looking to mine and make use of vast quantities of unstructured data.

While having a lot in common with traditional NAS—such as supporting NFS and SMB/CIFS protocols over Ethernet networks—there are a couple of key principles that define scale-out NAS. These are as follows:

- Horizontal scalability
- Single global namespace

Horizontal Scalability

Horizontal scalability is just a fancy term for *scale-out*. Let's quickly define the term. Consider an application that runs on a server. Adding more memory, more storage, or more CPU to the server that the application runs on is called *scale-up*, or *vertical scaling*. Adding more servers, each with CPU, memory, and storage, to the application is scaling out, or *horizontal scaling*.

The principle in the NAS world is the same. A NAS system that supports scale-up will allow you to add more memory and storage to it, which is different from scale-out, whereby more nodes/heads/controllers can be added. On the other hand, a scale-out NAS system will allow you to add more nodes to it, with each node bringing its own CPU, memory, and storage to the NAS system. These scale-out NAS systems are often referred to as *clusters* or *NAS clusters*.

NOTE A good way to think of horizontal scaling is spreading the workload over more nodes.

Scale-out NAS clusters support multiple nodes that can be dynamically added to the cluster. Each node has its own CPU, memory, and storage that contribute to the overall NAS cluster. A high-speed, low-latency, backend network is usually required so that read access to data stored on drives attached to different nodes isn't slow. 10 GB Ethernet and InfiniBand are popular backend interconnect technologies.

Writing software and filesystems that support scale-out is a tricky business, which is why not all NAS systems are scale-out.

Single Global Namespace

Typically, a scale-out NAS cluster will boast a single infinitely scalable filesystem. A filesystem is sometimes referred to as a *namespace*. All nodes in the scale-out NAS cluster run the filesystem and have full read/write access to it.

Because there is only one filesystem (namespace) and it is accessed by all nodes in the cluster, it can be referred to as a *single global namespace*. Having a single global namespace like this leads naturally to evenly balanced data placement on the backend, not only leading to massively parallel I/O operations, but also reducing drive and node hot spots.

Aside from being massive and global, these filesystems are designed to allow nodes to be seamlessly added and removed from the cluster. This can help immensely when addressing technology refresh, such as when replacing old hardware with new. For example, new nodes running newer hardware can be admitted to the cluster, and nodes running old versions of hardware can be evicted, all seamlessly, making hardware technology refresh way simpler than it has been with traditional NAS arrays.

Scale-out filesystems like these are often described as *distributed* and *cluster-aware*. Typically, when data is written to the scale-out NAS system, the data will be distributed across the entire backend.

Anyone who has managed large traditional NAS arrays with multiple filesystems that are locked to just two nodes knows what a management pain in the neck that is. A single global namespace goes a long, long way to simplifying management as well as scalability. Figure 7.5 shows a traditional two-head NAS with multiple filesystems on the backend, each accessible primarily by its owning/connected node. Figure 7.6 shows a small, five-node scale-out NAS with a single filesystem (global namespace) on the backend that is equally accessible by all nodes in the cluster.

FIGURE 7.5 Traditional two-node NAS with multiple backend filesystems

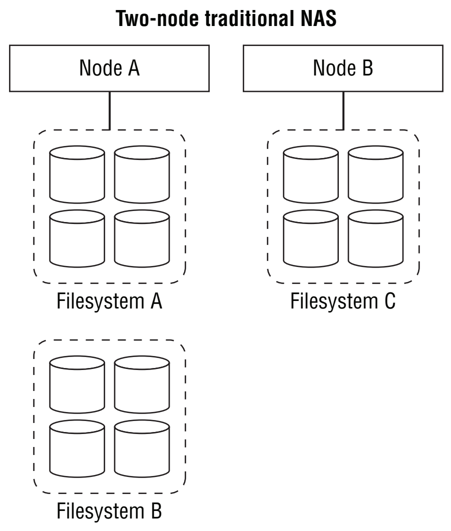
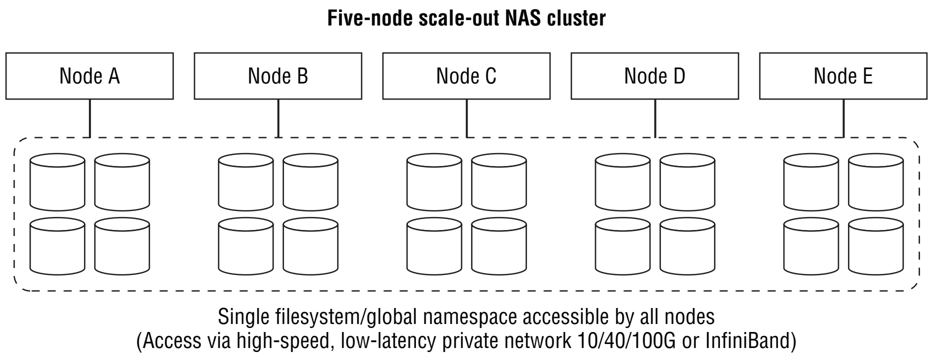


FIGURE 7.6 Five-node scale-out NAS with single global namespace


Five-node scale-out NAS cluster
Node A
Node B
Node C
Node D
Node E
Single filesystem/global namespace accessible by all nodes
(Access via high-speed, low-latency private network 10/40/100G or InfiniBand)

Generally speaking, scale-out architectures are preferred for large environments, as they prevent NAS sprawl. Instead of having to deploy many traditional NAS arrays, each with multiple filesystems that have to be managed separately, you can deploy a single scale-out NAS cluster running a single multi-petabyte-scale filesystem.

Other Scale-Out NAS Considerations

A couple of other matters are usually common to scale-out NAS. More often than not, scale-out NAS is built on top of commodity hardware. It is rare to see a scale-out NAS running ASICs and FPGAs. Also, scale-out NAS architectures often offer flexible protection schemes that work at the file level. For instance, individual files can be assigned different protection levels. For example, you might want to keep two copies of some files and three copies of some other files. Copies of files are usually kept on separate nodes in the cluster. Sometimes this is referred to as a form of network-level RAID or RAIN.

Redundant Array of Independent Nodes

RAIN is an acronym for *redundant array of independent nodes*, sometimes referred to as *reliable array of independent nodes*. If you've already read Chapter 4, "RAID: Keeping Your Data Safe," you know that RAID provides protection at the *drive level*. RAIN is very similar to RAID but provides protection at the *node level*. Basically, in a RAIN-protected system, you can lose an entire node in the cluster without losing data.

RAIN architectures usually provide protection via parity, error-correction code such as Reed-Solomon codes, or by simply making extra copies of files and objects.

RAIN is extremely popular in distributed computing and cloud-scale storage, including major object storage services such as Amazon S3 and the like. It is also a technology that allows you to build highly resilient solutions on top of commodity hardware.

Scale-out NAS clusters also commonly support more-exotic protocols such as parallel NFS (pNFS).

Gateway NAS Appliances

A *NAS gateway* is a traditional NAS front-end without its own native drives on the backend. Basically, a NAS gateway is the NAS heads (nodes/controllers) that run the NAS operating systems and file-sharing protocols, but without any of its own capacity on the backend. Instead of having its own drives, a NAS gateway connects to an external FC storage array for its capacity. This is shown in Figure 7.7.

The external storage can be third-party storage, and it can be either dedicated to the NAS gateway or shared with other devices. However, if the external storage is shared with other systems, it is common to dedicate specific drives to the NAS gateway, as shown in Figure 7.8.

FIGURE 7.7 NAS gateway connecting to FC storage array

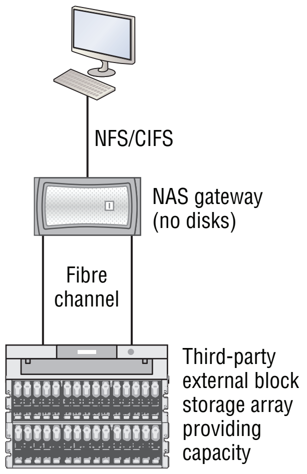
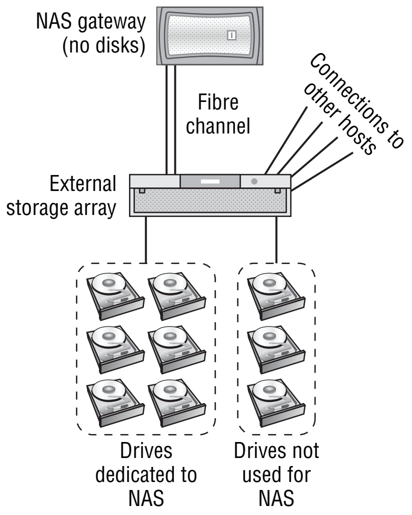


FIGURE 7.8 Dedicated NAS gateway drives on a shared external array



Beware of complexities in NAS gateway configurations, and make sure that you stick to your vendors' supported guidelines so that if things go wrong and you need to rely on the vendors, they will support you.

Obviously, a NAS client has no idea whether it is talking to a traditional NAS with its own captive storage backend, or whether it is talking to a NAS gateway.

Other NAS Array Goodness

As mentioned in Chapter 3, NAS arrays often provide important technologies, such as the following:

- Deduplication
- Compression
- Remote replication
- Snapshots
- Hypervisor integrations and offloads

NAS Performance

Aside from the usual storage culprits such as disk drive speed and cache sizing, there are a ton of other things that can significantly impact NAS performance. Let's take a minute to highlight some of these.

Network Impact on NAS Performance

As NAS technologies operate over IP networks, they are massively affected, positively and negatively, by the performance of the network. And as with all things storage related, latency is usually the most important performance characteristic. High latency causes irate users, and if it gets bad enough, applications can become useless to the point that they are considered *down*. So take this seriously!

Because of this, 10 GB Ethernet and dedicated LANs should be seriously considered. 10 GB Ethernet offers increased bandwidth, and dedicated networks remove congestion, which is a killer for latency. Also, if latency is a real concern and you cannot go with SAN or locally attached storage, you should seriously consider low-latency switches and at the very least keep the number of network hops to a minimum.

Jumbo frames can also improve performance. At a low level, communication between a client and a server occurs over Ethernet frames. And jumbo frames can reduce the number of frames sent between the client and server, improving overall network performance. When working with jumbo frames, make sure they are supported and enabled on all devices in the data path—NICs, switches, and NAS devices.

Link aggregation technologies, sometimes referred to as *bonding*, can also be helpful. Link aggregation can be applied at the host (client) in the form of NIC teaming, as well as in the network with technologies such as Link Aggregation Control Protocol (LACP). These technologies can help improve throughput as well as multipath failover.

NOTE *Link aggregation* is similar to ISL trunks/port channels in the FC world. The basic notion is combining multiple links between two switches into a single logical link with increased throughput and improved failover.

TCP window scaling can also impact performance, especially on a high-latency network such as a WAN. TCP window framing works by a server advertising how much unacknowledged data it can accept before it will stop accepting more data. A larger TCP window allows a server to accept more unacknowledged data, and this can significantly improve performance over high-latency networks such as WANs, where it can take a long time for data sent from one host to arrive at the destination. Think of the link between two hosts as a pipe; a larger TCP window allows you to fill up the pipe.

Non-Network-Related NAS Performance Influencers

Away from the network, NAS and file-based storage has a high-metadata overhead: the metadata-to-user data ratio is high. This means that waiting around for slow metadata operations to complete can drag NAS performance into the gutter. Therefore, some NAS designs, including scale-out NAS clusters, ensure that metadata is always stored on flash media for fast access.

Antivirus (AV) solutions can also impede NAS performance if the file-scanning overhead is too high. Make sure you follow your vendor's best practices when implementing AV solutions around a NAS deployment.

Implementing security in a NAS environment can also increase overhead and reduce performance. Early implementations of Kerberized NFSv4 in privacy mode (end-to-end encryption) added a significant CPU overhead to operations. However, modern, faster CPUs have vastly improved things in this area. You should still test the performance impact of things like this before implementing them.

Object Storage

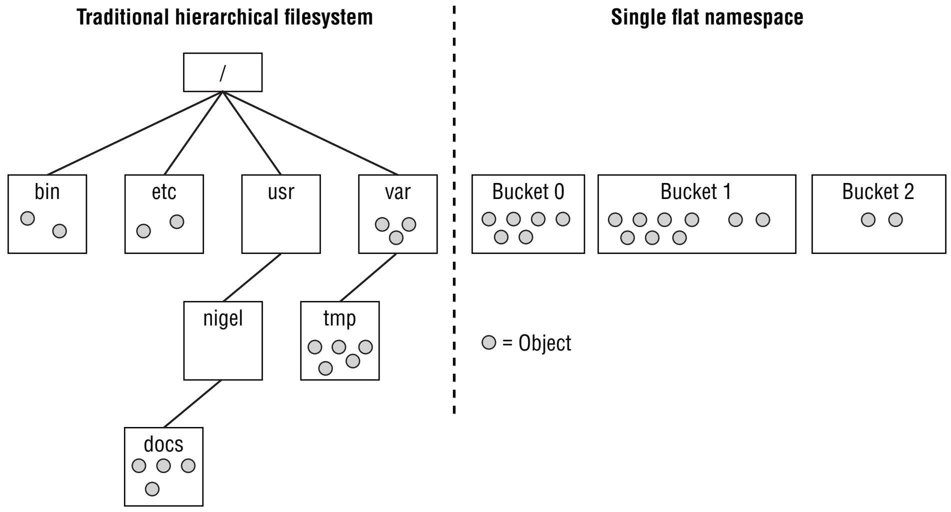
Let's start out with a quick high-level overview and then get a little deeper. *Object storage* requires a bit of a mindset change for some people. Keep in mind that it is intended for cloud-scale and cloud-use cases as well as designed to be accessed via RESTful APIs. You will examine all of this more closely soon.

If we were to compare object storage to SAN or NAS storage, it would probably be fair to say that object storage has more in common with NAS than SAN. This is mainly because objects are more like files than they are blocks. In fact, it could be said that in many cases files are objects, and objects are files. However, there are several key differences between an object storage device (OSD) and a NAS system.

First, objects don't always have human-friendly names. They are often identified by hideously long (for example, 64-bit) unique identifiers that are derived from the content of the object plus some arbitrary hashing scheme.

Next, objects are stored in a single, large, flat namespace. *Flat* means that there is no hierarchy or tree structure as there is with a traditional filesystem. This flat namespace is a key factor in the massive scalability inherent in object storage systems. Even scale-out NAS systems, with the massive petabyte-scale filesystems, limit the number of files in either the filesystem or individual directories within the filesystem. Figure 7.9 shows a traditional hierarchical filesystem compared to the flat namespace of an object store.

FIGURE 7.9 Filesystem hierarchy vs. flat object store



Another key difference between object storage and NAS is that object storage devices are not usually mounted over the network like a traditional NAS filesystem. Instead of being mounted and accessed via protocols such as NFS and SMB/CIFS, object storage devices are accessed via APIs such as REST, SOAP, and XAM. Quite often it will be an application that talks with the OSD rather than a human being.

NOTE The Storage Networking Industry Association (SNIA) developed the XAM API in an attempt to provide a standards-based interface for object stores and CAS systems, to avoid vendor lock-in. XAM is an acronym for eXtensible Access Method.

Let's take a moment to look at a really useful analogy comparing NAS with object storage. If NAS was compared to parking your car while out for a meal, you would have to personally drive your car into the parking garage, choose a level, and then choose a parking space. It would then be your responsibility to remember the level and space where you parked. Object storage is more like valet parking, where you drive your car up to the front of the restaurant, jump out, and exchange your car for a ticket, leaving the task of parking your car to the restaurant staff. While you're in the restaurant, you have no idea where your car is parked. All you care about is that when you're finished eating and socializing and want your car back, you can hand in your ticket, and your car will be returned exactly as you left it. With object storage, you (or your application) create some data and give it to the OSD in exchange for a unique object ID (OID). You (and your application) don't care where the object is stored, as long as it is protected and returned to you the next time you present your unique OID to the OSD. That should be pretty simple.

Object storage is not designed for high-performance and high-change requirements, nor is it designed for storage of structured data such as databases. This is because object storage often doesn't allow updates in place. It is also not necessarily the best choice for data that changes a lot. What it is great for is storage and retrieval of rich media and other Web 2.0 types of content such as photos, videos, audio, and other documents.

Now it is time to take a closer look at object storage.

Objects and Object IDs

Object storage devices are for storing objects. By *objects* we are really talking about files, and more often than not rich media files such as images, video, and audio files. Because they are usually rich media files, objects are often large, but they don't have to be. Although databases are also technically files, we wouldn't refer to database files as objects.

Every object stored in an object store is assigned its own unique object ID, known as an OID. The OID is usually a hideous 64-bit hash value that will mean nothing to you. However, it is absolutely vital.

Each object is stored in the object store, along with extensive metadata describing the object.

If we return to the valet parking analogy, we mentioned that we don't care where in the object store our objects are stored. Well, that is only true up to a point. You will want to consider policies that determine things such as what protection levels are applied to objects as well as what locations objects are available in. On a small scale, *locations* can refer to which nodes within the same data center an object is stored in; on a larger scale, it can refer to which country or continent a copy of an object is stored in. This is because object storage works at *cloud scale*, and object stores can, and do, span countries and continents. You may have nodes from the same object store in Australia, England, and the United States. Having copies of objects in each of these three locations leads naturally to protection (similar to RAID) as well as local access to data. If the nodes in Australia go down, you will still have copies of objects in England and the United States. Also, if you are based in the United States, accessing objects from the United States–based nodes will be faster than having to access remote copies in either Australia or England. That being said, object access will still not be fast compared to a SAN or NAS system.

When defining object protection policies, the more copies you keep, the better protected your objects are, but at the expense of space. If you don't stipulate exactly where additional copies of an object will be kept, the OSD will make sure that copies are kept in locations that provide the highest levels of availability. For instance, the OSD will not keep three copies on the same disks attached to the same node, as this would make the node and its disks a single point of failure.

Another consideration is the number and size of objects. Theoretically, some object storage devices support almost unlimited numbers of objects and unlimited sizes of objects. In reality, object stores with billions of objects are common. On the topic of object size, this depends on your choice of object storage technology, but objects can be very big. We're talking terabytes!

Metadata

If you thought metadata was important in NAS, wait until you see object storage. Extensive metadata is a hallmark of object-based storage.

Forget trying to remember in which directory, within a directory, within a directory, that you can't remember the name of anymore, you saved that file six weeks ago. Object storage is all about searching and indexing based on metadata. A common example is medical images such as X-ray and MRI images. These can be tagged with powerful metadata including things such as name of patient, age of patient, gender, ethnic background, height, weight, smoker, drinker, prescribed medication, existing medical conditions, and so on. All of these things can be searched and utilized. Another common example is traffic management and surveillance systems, where saved video images can be tagged with extensive metadata such as road type, time of day, time of year, weather, visibility, congestion metric, thumbnail view, and so on.

Metadata is stored with the object in standard key-value pairs. And generally speaking, an OSD will tag every object with basic metadata such as size, creation time, last modified time, and so on. After that, you or your application can define almost limitless custom metadata. The OSD will probably not understand this user-defined custom metadata, but this is not important. All that is important is that it exists and can be searched on and utilized by your applications.

Once the metadata is added to the object, the data payload and metadata are then stored together within the OSD, although the OSD may well chunk the object up for more-efficient storage on the backend.

API Access

As mentioned earlier, not all OSDs provide a NAS interface. Access to the OSD is often just via APIs such as RESTful HTTP, SOAP, or sometimes language-specific libraries and APIs such as Python, PHP, Java, and so on. This means that object storage is absolutely not a direct replacement for NAS.

A *RESTful HTTP API* means very simple `PUT/GET/POST/DELETE` operations that are common to HTTP and the Web. Examples of `GET` and `PUT` commands are shown next. The `GET` will list all containers in the `npoulton` account, whereas the `PUT` will create a new container called `uber_container`:

```
GET http://swift.nigelpoulton.com/v1/npoulton/
PUT http://swift.nigelpoulton.com/v1/npoulton/uber_container
```

These examples are a little oversimplified because requests should always require an authentication token. Therefore, the `PUT` command to create the container would probably look more like this:

```
curl -X PUT -H 'X-Auth-Token: AUTH_tk3…bc8c' http://swift.nigelpoulton.com/v1/
AUTH_admin/uber_container
```

Because objects in an object store can be manipulated by lightweight HTTP web services, each object can, and usually is, given its own unique URL by which it can be accessed. For example:

```
http://swift.nigelpoulton.com/v1/npoulton/uber_container/lily_in_a_tree.eps
```

Some object stores do provide a NAS interface, allowing them to be mounted over the network and accessed via the likes of NFS and CIFS.

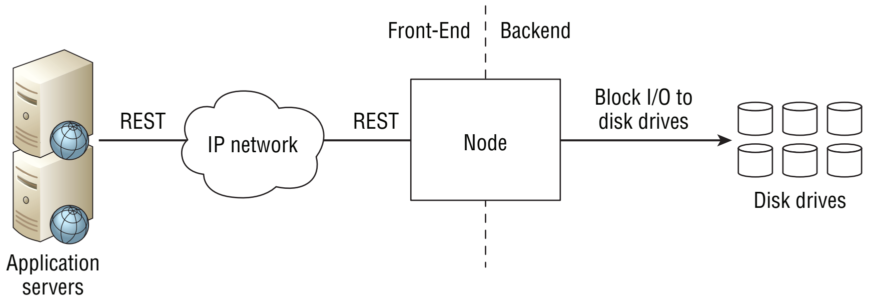
Architecture

Object storage is designed with the cloud in mind, including factors such as massive scalability, ultra-low cost, and relatively low performance. It is also designed to work with standard web APIs such as REST.

NOTE For a more detailed discussion and explanation of private, public, and hybrid cloud models, see Chapter 15, "Cloud Storage."

Architecturally speaking, object storage devices are pretty simple. They have a front end and a backend. The front end is where the clever stuff happens. It is effectively the broker between the clients that speak REST and SOAP APIs, and the backend that speaks lower-level block storage protocols. The front-end usually deals with the extensive metadata, too, and works over Ethernet networks running TCP/IP. There is also a backend private network used for node-to-node communication. Figure 7.10 shows a simple object storage architecture.

FIGURE 7.10 Simple object storage architecture



Unlike scale-out NAS technologies that run their backend over high-speed 10 GB and 40 GB Ethernet, or sometimes InfiniBand, the private networks in object stores are not high speed or low latency. In fact, they often have to traverse WANs and the Internet—so rather than high speed and low latency, they often operate over low-speed, high-latency links.

While the namespace can be carved into partitions, sometimes called *containers* or *buckets*, these containers and buckets cannot be nested. This is a major difference from traditional filesystems, where directories are nested within directories. Object store namespaces are flat. The namespace of an object store is also fully distributed, even in object stores that span the globe.

Public or Private Cloud

Object storage is relatively new and designed with the cloud in mind. Rather than building their own in-house object storage systems, many companies are opting to utilize cloud-based object storage services such as Amazon S3, Windows Azure, and Rackspace Cloud Files. These cloud-based storage services—known as *public cloud*—usually operate on a pay-as-you-go pricing model that allows companies to avoid the up-front capital costs involved in purchasing your own hardware and software, thus lowering the initial costs involved with deploying object-based storage.

With cloud-based models, you create an account with a provider, ship them your data—either via the Internet or by physically shipping them USB hard drives with your data on them—and then you start using the service. From that point on, you typically pay monthly, depending on how much capacity and bandwidth you use. You can access and manage your data via the APIs that your cloud storage provider supports, or via a control-panel–type tool they provide for you. Some cloud storage providers also provide, or partner with, content delivery network (CDN) providers to offer faster access to your data. CDNs speed up access to your data by placing web-based servers in strategic locations around the world that cache your data. For example, if your cloud storage provider has data centers in the United States and the United Kingdom, but you are operating out of Hong Kong, a CDN placed near Hong Kong that caches your data set will vastly improve your overall cloud storage experience.

The major alternative to public cloud is private cloud, and a popular private cloud option is OpenStack. OpenStack is a suite of cloud platforms that includes compute, network, storage, and management functionality. The major OpenStack storage offering is OpenStack Object Storage, commonly referred to by its code name OpenStack Swift. OpenStack software was initially a joint venture between Rackspace and NASA that was designed to reduce the costs of operating IT infrastructure by running infrastructure services as software that sits on top of cheap commodity servers, networking, and storage hardware. OpenStack is now backed by many of the traditional big IT companies such as HP, IBM, Dell, Red Hat, and Cisco and is gaining a lot of traction with companies looking to move away from purchasing expensive storage technologies from the traditional storage vendors.

OpenStack Swift is the technology that powers Rackspace Cloud Files and also underpins many of the other public cloud services available. It has extensive API access and offers all the major features of object storage, including authentication, containers, varying levels of redundancy, encryption, and more.

Security

As with all things that touch an IP network, especially the Internet, security is of paramount importance. For this reason, you will want to ensure that any object storage solution you choose to utilize mandates that all operations include a valid authorization token from your authorization subsystem. Fortunately, most do.

Object Store Options

There are tons of examples of object stores out there. Some are public cloud, some are private cloud, and some offer hybrid models. Let's take a brief look at some of them.

Amazon S3

If you thought Amazon.com was a book-selling company, you're still living in the '90s and are way behind the times. Amazon is a technology company and a massive, cloud-based, data storage company. *Amazon Simple Storage Service (S3)* is probably the biggest and best-known object storage system in the world. There are trillions of objects stored in Amazon S3. Yes, trillions! This means there are more objects in Amazon S3 than we think there are stars in our Milky Way galaxy. That being said, S3 still has a long way to go until it stores more objects than there are grains of sand on the earth!

The *Simple* in S3 is the key to its popularity and widespread implementation. You can sign up and play with it today—as you can with other public cloud object stores. Because it is web based, you can access it and store and retrieve data from it wherever you are on the planet, as long as you have an Internet connection.

You can store and retrieve as much data as your Internet connection and bank balance will allow. S3 uses a utility-style billing system, which is common to most cloud services. You can also apply policies to your objects so that they are deleted or archived to a cheaper, slower object archive called Amazon Glacier.

S3 refers to unique OIDs as *keys*¸ and it stores objects in *buckets*. When you store an object to S3, it will be stored and protected over several Amazon facilities before you receive notification that it has been successfully stored. Network-based check summing is also performed when you store and retrieve objects, to ensure that no corruption has occurred over the network.

OpenStack Swift

Swift is OpenStack's object storage offering. Swift can be used to build public or private object stores. Swift is open source software that allows you to build your own horizontally scalable OSD on top of commodity hardware. It has an authenticated RESTful HTTP API interface. Like most object stores, Swift can be manipulated with the common Linux `curl` command. You can start small and scale your Swift OSD to hundreds or even thousands of nodes.

Despite being open source, Swift is a mature and feature-rich object storage system. These features include extensive API access via REST, with bindings for Python, PHP, Java, and more. It also offers bulk imports, authentication, encryption, and object replication.

OpenStack is being positioned as a major operating platform for the cloud and offers way more than just Swift. In the storage space, OpenStack also offers Cinder for block storage. Outside of storage there are components for computing, networking, automation, and more. Check out `www.openstack.org` for more information.

Compliance Archives and Content-Addressable Storage

Many organizations are subject to government- and industry-enforced regulations that require them to keep immutable copies of data. *Immutable* means that once the data is created, you must be able to guarantee that it cannot be changed. Another industry term associated with immutability of data is write once, read many (WORM). In WORM technologies, you can write an object once and read it as many times as you wish, but you cannot update or change the object. A common example of WORM is CD-ROM technologies.

NOTE A well-known example of a data-retention regulatory requirement is the Sarbanes-Oxley Act of 2002, known affectionately as SOX. Others include HIPAA and PCI.

A particular branch of object store, known as *content-addressable storage (CAS)*, is popular as a compliance archive. In a CAS system, all objects are fingerprinted with a hash that determines the location of the object in the object store. The concept that the content of an object determines where it is located in the store could be a mindset change for many people, but it is at the very heart of how CAS works.

As an object is placed in the CAS object store, the contents of the object are read and then ran through a hashing system to generate a *fingerprint* that is assigned to the object. This fingerprint determines the location of the object in the CAS store. Here is the important part: if you change any part of that object, you change its fingerprint, which also changes its location in the store! When reading an object again in the future, its hash will be evaluated, and you will know whether the object has been changed.

For this reason, CAS is a real natural when it comes to compliance archiving, and many CAS systems are designed to enforce strict data immutability and guarantee that objects under a retention policy cannot be tampered with during their retention period.

NOTE This process of guaranteeing that an object has not changed since it was placed in the store—because its fingerprint and address have not changed—is known as *assurance* or sometimes *content authenticity.* Therefore, CAS systems are said to provide *content assurance* or *content authenticity*.

Another thing that CAS is a natural at is deduplication. If a new object is added to the store that is identical to an existing object, it will get an identical fingerprint and an identical location in the store. When this happens, a second copy of the object cannot be created and the object is naturally deduplicated. While this is a form of deduplication, it should be noted that it is more correctly referred to as *file-level single instancing* and is nowhere near as powerful as block-level deduplication. That said, it is still a nice side-benefit of CAS systems.

When considering a CAS system that uses hash functions, it is vital that strong hash functions, such as SHA-1, are used. If weak hashing functions are used, you increase the risk of *collisions,* which occur when two different objects return the same hash value. When this happens, the CAS system thinks the two objects are the same, and *you lose data!* This is because the CAS system thinks that it already has a copy of the new object and therefore doesn't need to store the new object being submitted to the store. The only ways to avoid this are to use strong hashing functions that negate the chance of collisions or to perform bit-for-bit comparisons each time a hash match is found. Both of these have an associated overhead. Stronger hash functions are computationally more intensive than weak hash functions, whereas bit-for-bit comparisons require backend storage operations to read the existing object. Bit-for-bit comparisons work by comparing every bit in the newly submitted object with every bit the object already stored. Fortunately, most modern CAS systems employ strong hashes such as SHA-1 that are extremely unlikely to generate collisions.

One thing that CAS is not great for is data that changes frequently. The overhead of continually changing the fingerprint and storage location for lots of objects that are frequently updated can be a real burden to CAS systems.

Object Storage Summary

Object storage is almost perfect for cloud-scale storage of data that doesn't change often and doesn't need particularly fast access. Storage of pictures and video files taken on mobile devices and uploaded to social networking sites and private cloud storage would be impossible with traditional storage technologies such as NAS and SAN. Object storage came marching to the rescue here!

Object stores also provide a neat solution to long-term data archives, especially compliance archives that have to store immutable copies of data.

Because they are inherently slow and not a good fit for data with a high rate of change, object stores won't completely take over the corporate data center. However, they already own the Internet. In corporate data centers, object stores will sit alongside NAS and SAN technologies for many years to come.

When looking at solutions for large amounts of unstructured data, object stores deserve close inspection. They also provide an opportunity to move away from traditional, vendor-based, prepackaged, off-the-shelf offerings and an opportunity to embrace commodity hardware with software-driven intelligence.

Chapter Essentials

NAS Arrays NAS arrays are purpose-built storage systems designed to provide shared file services over a shared IP network. They can be used to consolidate multiple general-purpose file servers such as Windows and Linux servers. They implement NFS and SMB/CIFS protocols and offer high performance, high availability, and advanced features such as snapshots and remote replication.

Traditional NAS Arrays Traditional NAS arrays are best suited for small and medium-sized businesses but struggle to meet the unstructured data demands of large businesses. They provide decent performance and advanced data services such as snapshots, remote replication, and data-efficiency technologies such as compression and deduplication. They do not scale well.

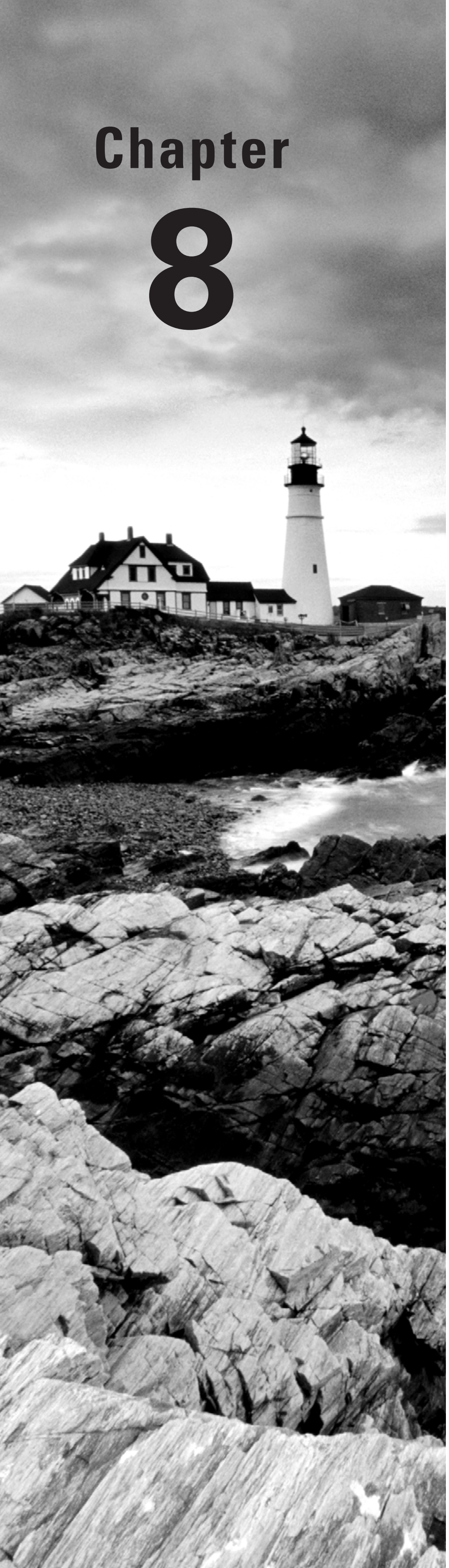
Scale-Out NAS Arrays Scale-out NAS arrays attempt to provide all that traditional NAS arrays provide, with the added benefit of massive scalability. They still use traditional file-serving protocols such as NFS and SMB/CIFS. They help avoid NAS sprawl in large environments and can be well suited for Big Data projects.

Object Storage Object storage is a new class of storage system designed for cloud-scale scalability. Objects are stored and retrieved from an object store via web-based APIs such as REST and SOAP. Each object can be tagged with extensive metadata that can be searched and indexed. Object storage is ideal for rich content data that does not change often and does not require high performance. It is popular in the public cloud model.

Content-Aware Storage Content-aware storage (CAS) is a form of object storage that is well suited as a compliance archive for providing data immutability to meet government and regulatory standards. In CAS systems, the content of an object determines its location within the object store, and any changes to the content of the object changes its location in the store. CAS in not well suited to data that changes frequently.

Summary

In this chapter we covered the topics of file storage and object storage. We started out by learning the major file serving protocols — SMB and NFS — and how they are used as well as what security features they have. We then learned about Network Attached Storage (NAS) arrays, compared them to local storage and SAN storage and explained some of the advanced and enterprise class features they often have. We also talked about some of the scalability limitations of traditional NAS arrays and how scale-out NAS solutions are addressing this limitation. Then we moved on to the exciting topic of object storage and explained how it differs from more traditional block and file storage solutions, and how these differences make it ideal for most cloud storage requirements. We talked about object storage architecture and how it is primarily accessed via APIs, as well as the importance of metadata in object storage.



Chapter 8

Replication Technologies

TOPICS COVERED IN THIS CHAPTER:

- ✓ **Synchronous replication**
- ✓ **Business continuity**
- ✓ **Asynchronous replication**
- ✓ **Array-based replication**
- ✓ **Database log shipping**
- ✓ **Logical volume manager snapshots**
- ✓ **Hypervisor replication and snapshots**
- ✓ **Multi-site replication topologies**
- ✓ **Array-based snapshots**
- ✓ **Array-based clones**

This chapter presents replication and snapshot technologies and how they fit into business continuity and disaster recovery planning and execution. You'll look at the types of remote replication technologies, such as synchronous and asynchronous, and the impact each has on network links required to support them. You'll also look at how replication and snapshot technologies are implemented at different layers in the stack, such as in the storage array, in the hypervisor, in a volume manager, and within applications and databases. Finally, you will see how to make sure applications and filesystems are consistent, and why consistency is important.

Business Continuity

Replication and snapshot technologies are a fundamental component of just about every *business continuity (BC)* plan. Business continuity is all about keeping the business running in the event of a disaster. Because we use the term *disaster,* business continuity is often referred to as *disaster recovery (DR)*. And when we use the term *disaster,* we mean any event that impacts the smooth-running business IT systems. Common examples of events that are considered disasters include the following:

- System failures and crashes
- Power failures
- Water leaks
- Natural disasters
- Human errors

One example is an approaching hurricane or major storm that will affect the site that is running your live business applications. As such a storm approaches, you will want to move these applications so they are running out of another site that will not be affected by the storm.

Another example is the failure of a large shared storage array—such things do happen—requiring you to bring affected applications back up on another array, potentially in another site.

All of these things require careful planning and plenty of practice!

NOTE While it is true that things don't go wrong too often in the storage world—especially when compared with the network world—it's fair to say that when things do go wrong in the storage world, they tend to go wrong in style! Although storage issues are few and far between, when they happen they tend to be big.

It's vital to understand that in today's world, most businesses are so reliant upon technology that it would be true to say that the technology *is* the business. No technology = no business! Unfortunately, it can often be quite hard to convince C-level management of this fact, even though they won't hesitate to come and shout at you when systems are down and the business is losing money!

Although 99 percent of businesses don't need to worry about hurricanes, many businesses have planned events that require them to move application services to an alternative site while maintenance work takes place. These planned events might include things such as annual power maintenance or other maintenance and engineering (M&E) works.

But business continuity is more than just making sure IT systems are up and running. It's all about making sure you can still effectively run the business. For example, it's no good having all storage and servers up and running in a remote DR site if people can't access those services or if performance is so poor that the business cannot effectively operate.

This chapter concentrates on the data replication components of business continuity and discusses volume snapshots. However, other key factors in a business continuity plan include backups, workplace recovery (desks for staff to work from if HQ is down), working from home, effective communications during the disaster, bringing servers and applications back online, and so on.

At the end of the day, everyone in IT has a responsibility to the business, as well as the customers of the business, to ensure that the business can operate in the event of a disaster. If IT infrastructure goes down and you do not have a plan of how to get the business back up and running, you risk losing your job, other people's jobs, and potentially the entire business. So, no pressure!

Having a plan of how to get the business back up and running doesn't just mean scribbling a few ideas on a notepad somewhere. We're talking about proper plans and proper practice runs (sometimes called *rehearsals*). Rehearsals are absolutely vital in managing effective and reliable business continuity. Test your plans, learn from the things that didn't go to plan—and trust me, things never go quite according to plan—and then test again and again! You absolutely do not want to be testing your BC plans for the first time in the heat of a severity 1 disaster!

It's also important to understand the business importance of your applications. Mature organizations have all their applications mapped against priorities or tiers. These priorities are then used as part of the decision-making process when determining the order in which applications are recovered in the event of a disaster. For example, all priority 1 applications might be brought back online first, and you might have agreed with management that you will have all priority 1 applications back up and running within 2 hours. Priority 3 applications will be less important to the business, and you might have agreed with management that these will be back up and running within 10 hours of the disaster. Mature organizations rehearse their BC plans regularly—this cannot be stressed enough!

TIP

Do not make the mistake of thinking that systems used internally by the IT department aren't as important as priority 1, mission-critical, line-of-business applications in the event of a disaster. Your chances of being able to recover mission-critical applications will be severely reduced if you cannot log on to the systems required to recover your mission-critical applications. IT systems such as remote-access gateways that allow staff to log on remotely and management servers to run scripts and tools are important and often need to be up before you can even start recovering business applications. You can't run scripts to start business applications if you can't get logged on in the first place!

TIP

Some infrastructure-type applications that often get overlooked include email and corporate chat tools, which can be vital in coordinating efforts in the event of a disaster. During a disaster, communication is often the number one priority. There is no point in everyone in IT busily working on recovering from the disaster if everyone is working in different directions.

OK, that covers the background and the important basics. Before you start looking in more detail at the technologies underpinning it all, let's take a quick look at some of the language related to business continuity:

Recovery Point Objective (RPO) This is the point in time to which a service is restored/recovered. In other words, if you have an RPO of 15 minutes, you need to be able to recover an application or service to the state it was in no more than 15 minutes before the disaster occurred. To do that, you could arrange for backups to be taken every 15 minutes. For example, a backup of a database taken at 8:30 a.m. can recover the state of the database to what it was at the time of the backup (8:30 a.m.). However, if a disaster strikes at 8:44 a.m., the backup cannot recover the database to how it was between 8:30 and 8:44. The backup can recover the application only to the state it was in at exactly 8:30—the exact time the backup was taken. As a note, backups are vital, but do not normally provide the best RPOs. For a more granular RPO, frequent snapshots of continuous data protection (CDP) products are often a better choice.

Recovery Time Objective (RTO) This refers to how long it takes to get an application or service back up and running. If we go with the same disaster at 8:44, and it takes you 2 hours to recover the database to the state it was in 15 minutes prior to the disaster, then your RTO should be no less than 2 hours. In terms of RTO, this is a period of time you and the business manager pre-agree that it will take to recover an application or service. So if you know that at best it will take you 2 hours to recover a service, you should agree to a more realistic RTO of something like 3 hours. Of course, if you encounter a severe unexpected disaster and you have to recover 20 applications, each with an RTO of 2 hours, you will probably not be able to recover them all within the 2 hours.

Service-Level Agreement (SLA) An SLA is a level of service that you agree to maintain with a customer or business unit, and you will see SLAs everywhere. You might have an SLA on a high-performance storage array that it will provide an I/O response time of less than 30 ms. In the arena of business continuity, SLAs tend to incorporate RPO and RTO. An important point to note about SLAs and RTOs is that although you might be able to recover email, finance systems, trading, or reporting systems within 2 hours, you will probably not be able to recover all of them in 2 hours if they are all affected by the same disaster. Put another way, you can recover email *or* finance systems *or* trading systems within 2 hours, but you cannot recover email *and* finance *and* trading within the same 2 hours! Such things are vital for business management to understand well in advance of any disasters.

Services When talking about BC, it is important to talk in terms of services or applications, meaning a collection of servers and other technology resources. For example, the business will not care if you can recover all your front-end web servers if you haven't recovered any of the required backend database servers that the web servers need to talk to. Protection and recovery should be thought of and performed in terms of services supplied to the business.

Invoking DR Invoking DR usually means bringing an application or service up at a remote site. For example, flaky hardware on a database server that causes the database server to be up and down would probably cause you to invoke DR for that application/service by moving all required components over to standby systems in your DR site. It is possible to invoke DR for a single server, an entire application, an entire service, or even an entire estate. Invoking DR for an entire estate is usually no mean feat!

Reduced Resiliency Reduced resiliency, in BC terms, usually refers to a loss of the ability to invoke DR or a loss of high availability. An example of losing the ability to invoke DR is losing power at your DR site, meaning you cannot recover services to that site if you experience issues at your main production site. An example of loss of high availability is losing one SAN fabric; under normal operating circumstances, you will probably run dual redundant fabrics, and losing one of those fabrics will reduce your resiliency. The thing to note about reduced resiliency is that your business applications are not usually impacted; you are just running at higher risk that a failure could incur more downtime than you can tolerate.

High Availability (HA) HA refers to IT designing and deploying solutions in a resilient manner. A common example in the storage world is deploying dual independent fabrics, with each server connected to both fabrics, so that if one fabric fails, service can continue over the surviving fabric. Only if both fabrics fail simultaneously will DR have to be invoked. A major goal of HA is to prevent having to invoke DR unless absolutely necessary. This is because HA usually doesn't require any service outage. In the dual fabric example, when one fabric fails, the other immediately takes over without any service outage. This is usually not the case when invoking DR. Invoking DR nearly always involves a short amount of service downtime.

TIP

When it comes to business continuity, there is a lot to consider and a lot to be missed if you don't test your business continuity plans regularly. Test them regularly!

Replication

Replication is a core component to any BC and DR strategy. However, it is only one component. Replication must be combined with effective backups of data and a whole lot more in order to form a viable business continuity strategy.

NOTE In this chapter, the term *replication* refers specifically to remote replication—copying data from a primary site to a remote secondary site. Sometimes this remote replication technology is referred to as *mirroring,* whereby you create a remote mirror.

You use remote replication to keep copies of your data at one or more remote sites. Two major uses for these remote replicas are as follows:

- Business continuity
- Reporting and testing/development

In the case of BC, remote replicas of data can often be used to recover applications and services in the event of outages such as crashed systems, failed power, water leaks, and natural disasters. In the case of reporting and testing/development use cases, remote replicas are often used as read-only copies of data that you run reports against. An advantage of using the remote replicas to run reports against is that any workload imposed on the remote replica, such as heavy read workload as a result of the report, is isolated to the remote replica and has no performance impact on the live production copy of the data.

Replication can be performed at various layers in the stack. These are the most common layers where it happens:

- Application/database layer
- Host layer (logical volume manager based in the Linux world)
- Storage layer (array based)

You'll take a look at each of these, but first let's look at a couple of important concepts that relate to replication in general, no matter where in the stack it is implemented.

WARNING Although replication technologies are great at recovering from natural disasters, they're usually no good at protecting your data from logical corruption. Logical corruptions such as viruses will be faithfully replicated. Replication also won't help you if data is accidentally deleted. Again, deletions will be faithfully replicated to your remote replicas, ensuring that data is wiped from your remote site, too.

Synchronous Replication

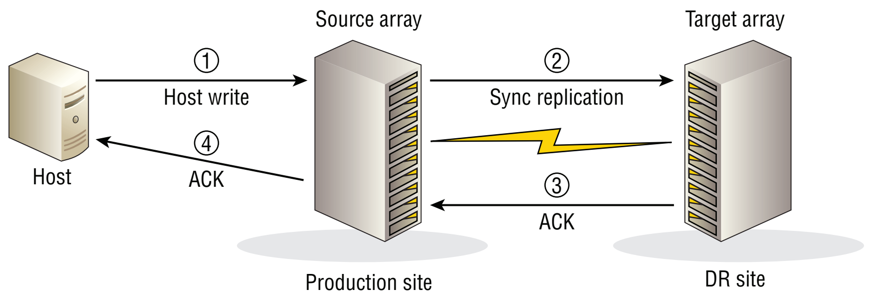
Synchronous replication technologies guarantee the remote replica to be up-to-date with the source copy of the data, which we sometimes call *zero data loss*. But to achieve this, synchronous replication technologies have a negative impact on performance. The reason for this is that all writes to a volume that is synchronously replicated are committed to both the source and the remote replica before the write is considered complete. This ensures that the source and target are always in sync, but it also means that the write data must travel all the way to the remote replica, and then an acknowledgment must return from the remote replica, before the write is considered complete. This round-trip to the remote replica and back can take a fairly significant amount of time (several milliseconds) and is dependent on the distance between the source and the target.

NOTE

Synchronous replication offers zero data loss, which gives an RPO of zero. However, the consistency of the replica might require you to perform additional work to bring it into a consistent state. This extra work might have an impact on the RTO that you agree upon with the business management.

Figure 8.1 shows a storyboard of array-based synchronous remote replication.

FIGURE 8.1 Synchronous replication storyboard



Replication Distances and Latency

Replication latency is impacted by several factors, including the technology and equipment involved, the number of routers that have to be traversed en route, and the distance between the source and target. A good rule of thumb when it comes to latency and distance is to assume about 1 ms of latency for every 100 miles between the source and target. Of course, if you have to traverse a lot of routers as part of that journey, your latency will increase as each router has to process your packets.

Network latency can be (somewhat crudely) tested by using the `ping` command. The following `ping` command shows the round-trip time from a machine in London to a machine in New York over a corporate Multiprotocol Label Switching (MPLS) network. The distance between London and New York is roughly 3,500 miles, and the average round-trip time (RTT) is 77 ms— about 38.5 ms there, and another 38.5 ms back. Assuming a latency of 1 ms for every 100 miles, this gives us a distance between the London-based system and New York–based system of 3,850 miles. The math is 77 ms / 2 = 38.5 ms (because the `ping` command is telling us the time it takes for a packet to get from London to New York and then back again). Then we take 38.5 ms and multiply by 100 to convert the milliseconds into miles, and we get 3,850 miles.

The names and IP addresses shown in the following output have been modified so they do not represent real names and IP addresses:

```
C:\users\nigelpoulton>ping nydc01

Pinging nydc01.company-internal-network.com [10.11.12.13] with 32 bytes of data:
Reply from 10.11.12.13: bytes=32 time=77ms TTL=52
Reply from 10.11.12.13: bytes=32 time=76ms TTL=52
Reply from 10.11.12.13: bytes=32 time=77ms TTL=52
Reply from 10.11.12.13: bytes=32 time=77ms TTL=52

Ping statistics for 10.11.12.13:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 76ms, Maximum = 77ms, Average = 77ms
```

Generally speaking, 100 miles is the upper limit on distances covered by synchronous replication technologies. Less than 100 miles is preferred. Either way, the key groups to engage when making decisions like this are as follows:

- Storage vendor
- Application vendor
- Network team

Your storage vendor and application vendors will have best practices for configuring and using replication, and you will definitely want to abide by their best practices. Your network should be able to provide accurate data about the network that your replication traffic will be transported over. Don't try to do it all on your own and hope for the best. Do your research and talk to the right people!

Replication Link Considerations

When it comes to replication, the transport network for your replication traffic is vital. A flaky network link, one that is constantly up and down, will have a major impact on your replication solution.

Synchronous replication solutions tend to be the worst affected by flaky network links. For example, in extreme cases, an unreliable network link can prevent updates to an application from occurring. This can be the case if the volumes that are being replicated are configured to prevent writes if the replication link is down; you may have certain applications and use cases where it is not acceptable for the source and target volumes to be out of sync. In the real world, 99.9 percent of businesses choose not to stop writes while the network is down. However, if you don't stop writes to the source volume when the replication link is down, the target volume will quickly become out of sync. And if you have an SLA against those volumes that states they will have an RPO of zero—meaning you can recover the application or service to the exact time of the outage—then you will be in breach of your SLA.

But problems with synchronous replication can be caused by factors other than links being down. A slow link, or one that's dropping a lot of packets, can slow the write response to source volumes because new writes to the volume have to be replicated to the remote system before being acknowledged back to the host/application as complete. Any delay in the replication process then has a direct impact on the performance of the system.

NOTE

Synchronous replication configurations always require robust and reliable network links. But if you happen to have applications and use cases requiring you to fence (stop writing to) volumes when the replication link is down, it is even more important that the network circuits that support your replication links are robust. In these situations, multiple network links in a highly available configuration may be required.

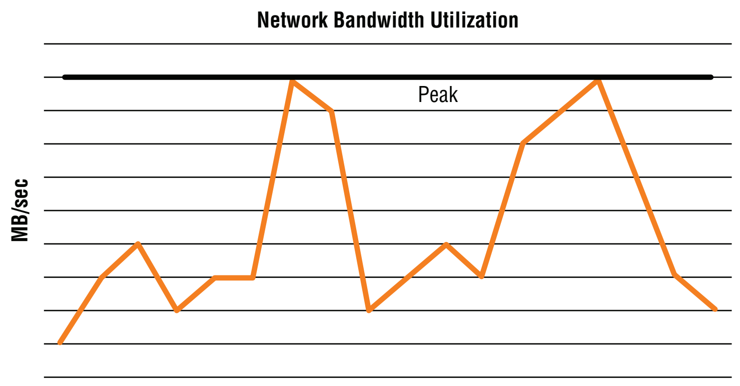
Because the replication network is so important in the context of synchronous replication, it is quite common to deploy multiple diverse replication networks between your source and target sites. Depending on the technologies in use, these diverse replication networks—diverse routes often via diverse telecom providers—can work in either an active/active or active/standby configuration. The design principle is that if one replication network fails, the other can handle the replication traffic with minimum disruption.

The sizing of your replication network—sometimes called *network link*—is also of paramount importance in synchronous replication configurations. If you get the sizing wrong, you will either be paying way over the odds or find yourself in the uncomfortable position of not being able to meet your RPO SLAs. An important design principle for synchronous replication configuration is sizing network bandwidth for peak traffic, including bursts. For example, if during high bursts of traffic the replication network is saturated and delay is introduced to the replication process, additional latency will be added to each write transaction while the source volume waits for the write transaction to be confirmed as complete at the target array, at the other end of the replication link.

Figure 8.2 shows network bandwidth utilization. If this link is to support a synchronous replication configuration, it will need to be sized to cope with the peak bursts of traffic.

Another important point to note, especially on shared IP networks, is that replication traffic can consume bandwidth and reduce performance of other applications using the shared network. Make sure you work with the network guys to put *policing* or other forms of network bandwidth throttling in place.

FIGURE 8.2 Synchronous remote replication bandwidth requirements



Asynchronous Replication

In many design respects, asynchronous replication is the opposite of synchronous. While synchronous replication offers zero data loss (RPO of zero) but adds latency to write transactions, asynchronous replication adds zero latency to write transactions, but will almost always lose data if you ever need to recover services from your remote replicas.

The reason that asynchronous replication adds no overhead to write transactions is that all write transactions are signaled as complete as soon as they are committed to the source volume. There is no required wait until the transaction is sent across the replication network to the target systems and then signaled back as complete. This process of copying the transaction over the replication network to the target is performed asynchronously, after the transaction is committed to the source. Of course, although this removes the performance penalty of synchronous remote replication, it introduces the potential for data loss. This potential for data loss occurs because the target volume will almost always be lagging behind the source volume. Exactly how far behind depends on the technology involved. It is not uncommon with some technologies for the target volume to be as far as 15 minutes or more behind the source—meaning that you could lose up to 15 minutes' worth of data if you lose your source volumes and have to recover services from the target volumes.

Another advantage of asynchronous replication technologies is that they allow the distances between your source and target volumes (or arrays) to be far larger than with synchronous replication technologies. This is because the RTT between the source and target is less important. After all, write performance is no longer affected by the latency of the replication network. In theory, distance between source and target volumes can be almost limitless. However, in practice, many technologies struggle over distances of more than 3,000 miles. This means you may struggle to replicate your data from an office in Sydney, Australia, back to a data center in New York (a distance of approximately 10,000 miles). In scenarios like this, it's more common to have a data center in the Asia-Pacific region that acts as a regional hub that all offices in the Asia-Pacific region replicate to.

There is also no requirement to spec network connections for asynchronous replication configurations to cope with peak demand. Therefore, network links for asynchronous replication can be cheaper, which can be a great cost saving.

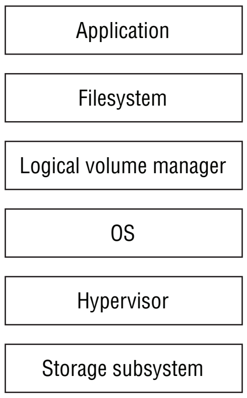
Of course, all of this is of use only if your services, applications, and users can cope with the potential of losing data in the event of a disaster.

NOTE

The term *invoke DR* (disaster recovery) refers to bringing an application up by using its replicated storage. The process normally involves failing the entire application over to a remote standby system. The process is usually disruptive, involves a lot of people, and can take up to several hours if there are a lot of applications and services that need failing over to DR.

That is enough about synchronous and asynchronous replication for now. Let's take a look at how replication is performed at the different layers in the stack, starting with the application layer. Figure 8.3 shows some components of the storage technology stack with the application layer sitting on the top. As we talk about how replication is performed at different layers in the stack, we'll start out at the top with the application layer, and work our way down the stack to the lower layers.

FIGURE 8.3 Some of the major layers of the storage stack



Application-Layer Replication

Applications sit on top of the storage stack. They are the highest level in the stack, above logical volume managers, hypervisors, and storage arrays.

Application-layer replication performs all the hard work of replication, at the application server. There is no offloading of replication tasks and replication management to the storage array. This hard work consumes CPU and memory resources of the application server, and although this was once seen as a big no-no, the recent advancements in CPU technology and the massive amounts of memory common in most of today's servers means that this hard work of replication can often be easily handled by the application server.

The major advantage of application-layer replication is undoubtedly application awareness, or application intelligence. The term *application-aware replication* means that the replication technology is tightly integrated with the application, often as a component of the application itself. And this tight integration of the replication engine and the application produces replicas that are in an *application-consistent state,* meaning that the replicas are perfect for quickly and effectively recovering the application. These kinds of *application-consistent replicas* are in stark contrast to *application-unaware replicas*—those that are not performed at the application layer or that are without any integration with the application—which are often in an inconsistent state and can require additional work in order for the application to be successfully recovered from them.

Popular examples of application-layer replication include technologies such as Oracle Data Guard and the native replication technologies that come with Microsoft applications such as Microsoft SQL Server and Microsoft Exchange Server.

Let's look at a couple of popular application-based replication technologies:

- Oracle Data Guard
- Microsoft Exchange Server

By the way, both of these replicate over IP.

Oracle Data Guard

Oracle databases can be replicated via storage array–based replication technologies or Oracle Data Guard, Oracle's own application-layer remote-protection suite. Both types of replication are extensively used in the real world, but there has recently been a shift toward using Data Guard more than array-based replication. Let's take a quick look at Data Guard.

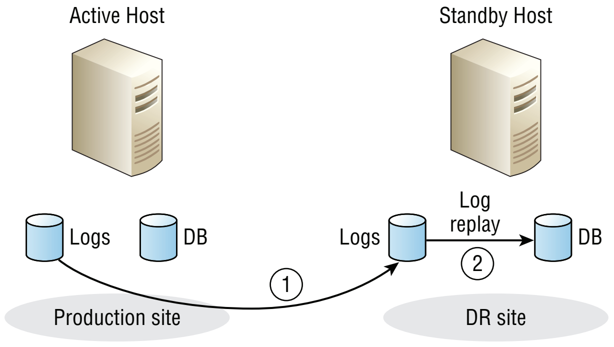
Oracle Data Guard, sometimes shortened to DG, is an application-based remote replication technology. Because it is written by Oracle specifically for Oracle, it has a lot of application awareness that array-based replication doesn't have. The net result is that Data Guard replicas are transactionally consistent.

In a DG configuration, a production database is known as the *primary*, and each remote disaster recovery copy of the database is known as a *standby*. There can be multiple standby copies of a primary database, but only one primary database in a DG configuration. Standby databases can, and should, be located in geographically remote locations so that if the site that hosts the primary database fails, standby databases are not affected and can be transitioned to primary.

DG is a form of log-shipping technology; redo logs from the primary are shipped to the standby databases, where they are applied to ensure transactionally consistent standby databases. Figure 8.4 shows a simple log-shipping configuration.

Application-layer replication can also be more bandwidth efficient than replication performed at lower layers in the stack, such as the array. For example, when replicating databases, array-based replication technologies have to replicate all changes to both the database and log file volumes. By comparison, application-layer replication based on log-shipping technology replicates changes to only the log files. Application-layer replication can also be more bandwidth efficient than array-based replication that is based on snapshot technologies. Snapshot-based replication technologies tend to replicate in units of extents, which can be far larger than the changed data they are required to replicate. For example, a 4 KB change to a 1 MB extent will require the entire 1 MB extent to be replicated.

FIGURE 8.4 Log-shipping application-based replication



Data Guard supports two types of standby databases:

Physical A physical standby database is ensured to be an exact physical copy of the primary database. This includes an identical on-disk block layout and an identical database configuration including row-for-row exactness, meaning that the standby will have the same ROWIDs and other information as the primary. This type of standby works by shipping the redo logs from the primary database and applying them directly to the standby database as redo transactions. Physical standby databases can be mounted as read-only while in a DG configuration.

Logical A logical standby database contains exactly the same data as the primary database, but the on-disk structures and database layout will be different. This is because the redo logs are converted into SQL statements that are then executed against the standby database. This difference allows the logical standby database to be more flexible than the physical standby database, resulting in it being able to be used for more than just DR and read-only (R/O) access.

Both standby database types are guaranteed to be transactionally consistent. And both perform data integrity checks at the standby DB before the replicated data is committed.

Oracle Data Guard also allows for both switchover and failover. *Switchover* is the process of manually transitioning a standby database from the standby role to the primary role. Usually associated with planned maintenance tasks, switchover guarantees zero data loss. *Failover* is usually an automatic process that occurs when the primary database fails. Failover can also be configured to guarantee zero data loss.

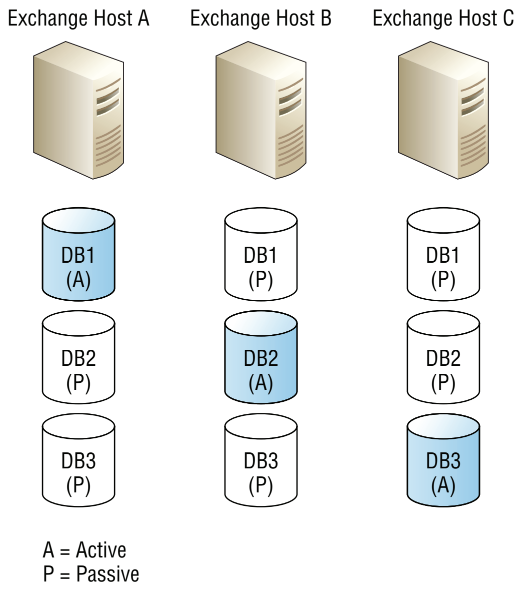
Microsoft Exchange Server Replication

Microsoft Exchange Server is another popular application that has its own native replication engine. You can do array-based replication, or you can do Exchange native replication. The choice is yours. Recently, there has been a real push toward using Exchange native replication rather than array-based replication. Both work, but Exchange's own native application-layer replication tends to be cheaper and has more application awareness. You also don't need to involve the storage team when failing over. On the downside, though, it doesn't support synchronous replication.

Like Oracle Data Guard and many other application-layer replication technologies that are database related, Exchange replication is based on an active/passive architecture that allows you to have more than one standby copy of a database. It also utilizes log shipping; you ship the transaction logs to a remote Exchange server where they are applied. Like most application-based replication technologies, it supports both switchover and failover.

As of Exchange 2010, the fundamental unit of replication is the data availability group, or DAG for short. A DAG is a collection of Exchange Servers that host active and standby copies of mailbox databases, as shown in Figure 8.5.

FIGURE 8.5 Microsoft Exchange data availability groups



In Figure 8.5, any server in the DAG can host a copy of a mailbox database from any other server in the DAG. Each server in the DAG must have its own copy of any mailbox database that it is configured to be active or standby for. Having a single copy of a database on shared storage that is seen and shared by multiple mailbox servers is not allowed!

Within the DAG, recovery is performed at the database level and can be configured to be automatic. Because recovery is performed at the database level, individual mailbox databases can be failed over or switched over, and you do not have to fail over or switch over at the server level (which would affect all databases on that server).

Also, Exchange replication uses Microsoft failover clustering behind the scenes, though this is not obvious from regular day-to-day operations.

NOTE With the introduction of DAGs in Microsoft Exchange Server 2010, replication no longer operates over SMB on TCP port 445. This can make life unbelievably simpler for people who work for organizations that are paranoid about security. And what organizations aren't paranoid about security these days?

Exchange Server 2010 and 2013 both provide a Synchronous Replication API that third-party storage vendors can tap into so they can provide synchronous replication capabilities to Exchange 2010 and 2013 DAGs. They effectively replace the native log-shipping replication engine with their own array-based synchronous block–based replication.

There are plenty of other database- and application-based replication technologies, but the two mentioned here—Oracle Data Guard and Microsoft Exchange Server—provide good examples of how they work. Some of the key factors in their favor include the following:

- Cost
- Application consistency
- Simplicity

Logical Volume Manager–Based Replication

Volume managers exist further down the technology stack than applications, but higher in the stack than hypervisors and storage arrays.

Host-based volume managers, such as Linux LVM, are capable of remote replication by using long-distance LVM mirrors. On the Linux LVM front, most people still consider this both a poor-man's replication technology and somewhat risky for production environments. That is fine for labs and maybe development environments, but it is not a technology to bet the business on. That said, it can be useful, and in the case of Linux LVM, it's free.

NOTE Because LVM is agnostic about the underlying storage technology, many companies have utilized LVM mirroring to perform storage migrations, whereby the storage presented to a server is being changed from one vendor to another, and the host stays the same. Some of these migrations have been performed when the server is running but applications and databases are shut down. However, at times LVM mirroring has been successfully used to migrate lightly used databases to new storage arrays. The latter should be done with extreme care, as there is no guarantee it will work.

In the Windows world, third-party products that are implemented as filter drivers above the volume management layer are often used to provide server-based replication. A common example, and one that has been around for some time now, is Double-Take.

Hypervisor-Based Replication

Hypervisors exist further down the technology stack than applications and logical volume managers, but higher in the stack than storage arrays.

As hypervisor technologies mature, they are starting to offer more and more advanced features natively. One of these features is replication. Although sometimes seen as a poor-man's replication choice, hypervisor-based replication technologies are maturing at a fast pace and have some potential advantages over array-based replication technologies—not to mention lower costs.

So far, interest in hypervisor-based replication technologies is strongest in small and medium businesses (SMB), but it is definitely worth a look even if you consider your company to be more than SMB. VMware vSphere Replication is currently the most advanced hypervisor replication technology, so let's take a look at it as an example.

vSphere Replication

VMware vSphere now offers a native hypervisor-level replication engine that can be leveraged by Site Recovery Manager (SRM). It goes by the name of *vSphere Replication*.

A couple of good things about vSphere Replication include reduced costs and a simplified configuration. Both of these factors are important in 99 percent of IT environments. Costs can be reduced by not having to purchase expensive array-based replication licenses, effectively bringing the advanced features of SRM replication to more people. vSphere Replication also brings a heap of flexibility to the storage layer by allowing storage platform independence and reducing any potential vendor lock-in at the storage layer. Basically, you can use any storage underneath, and your choice of storage technology does not have to be the same at the source and target.

WARNING

Be careful about using any storage and not making the effort of matching storage at the source and target. Although it reduces costs and allows more-flexible replication configurations, not matching the source and target storage has the potential to increase some of the risks surrounding the configuration. These risks include availability and performance, including potentially poor performance if you ever have to run your business on lower-performance storage in the event of a disaster.

vSphere Replication is an IP-based replication technology that works at the virtual machine disk (VMDK) level. Working at the VMDK level allows you to configure replication at a very granular level—more granular than even at the virtual machine level. For example, imagine you have a VM with four virtual disks as follows:

VMDK 1: OS partition

VMDK 2: Page-file partition

VMDK 3: Database partition

VMDK 4: Scratch partition

Now imagine you have a skinny network pipe between your data centers and want to make sure you replicate only the data that you will need in the event of a disaster. You can select to replicate only VMDK 1 and VMDK 3, and not to replicate the page file and scratch VMDK files. This could significantly ease the load placed on the replication network between sites.

On the efficiency front, vSphere Replication is intelligent enough to only replicate changed blocks for each VMDK, using a similar technology to the changed block tracking (CBT) engine used for incremental backups.

vSphere Replication also exists above the storage layer in the ESX stack, meaning that you are free to use any storage you want as the underlying storage—so long as it exists on VMware's hardware compatibility list (HCL). This is the opposite of when using array-based replication technologies that usually come with strict rules about which underlying storage platforms can replicate with each other. (Quite often these restrict you to the same vendor and same model for both the source and target.) For example, in a vSphere Replication configuration, your primary copy of a VM could be running on an iSCSI LUN from a Dell EqualLogic iSCSI storage array, but your remote offline VM might sit on a datastore on a local disk!

vSphere Replication replicates powered-on VMs (there is not much to replicate if the primary VM is powered off) from one ESXi host to another. These ESXi hosts can be in the same cluster in the same site or in different clusters at remote sites. Also, the ESXi host-based agent is installed by default, meaning no VM-based agents are required in order to use vSphere Replication.

vSphere Replication is also application aware on the Microsoft front, making API calls to the Microsoft Volume Shadow Copy Service (VSS) for the likes of Microsoft Exchange Server and Microsoft SQL Server, ensuring remote replicas are consistent and can safely and easily be recovered from.

As you can see in Exercise 8.1, standard vCenter tools are used to configure vSphere Replication as well as VM recovery. RPOs can also be configured for anywhere between a minimum of 15 minutes to a maximum of 24 hours.

EXERCISE 8.1

Configuring vSphere Replication

This exercise highlights the steps required to configure vSphere Replication (VR) via the vSphere Web Client. As a prerequisite, this example assumes that you have downloaded and are running the vSphere Replication Manager (VRM) virtual appliance and have vSphere Replication traffic enabled on a VMkernel NIC.

1. Within the vCenter Web Client, locate the powered-on VM that you want to replicate. Right-click the VM and choose All vSphere Replication Actions ➪ Configure Replication.

EXERCISE 8.1 *(continued)*

It is possible to configure vSphere Replication with just a single site configured in vCenter. Obviously, this kind of replication won't protect you from site failures, but it is good for practicing in your lab or for protecting against loss of datastores.

2. Choose the site you want to replicate to and click Next.
3. Choose a target datastore to replicate to (if you have only a single site, make sure the target datastore is a different datastore than the one your VM is currently in). Click Next.
4. Use the slider bar to choose an RPO between 15 minutes and 24 hours. Smaller is usually better.
5. If supported by your Guest OS, you can also choose to quiesce the VM.
6. Click Next.
7. Review your choices and click Finish.

vSphere will now make the required changes—such as injecting a vSCSI driver into the guest to manage changed block tracking—and then perform an Initial Full Sync. After the Initial Full Sync is complete, only changed blocks will be replicated.

In its current incarnation, the smallest RPO you can configure with vSphere Replication is 15 minutes. This will be fine for most use cases, but not all.

Despite its many attractive features, vSphere Replication is host based and therefore consumes host-based CPU and memory. Not always the end of the world, but in a world that demands efficient resource utilization on a platform that boasts massive server consolidation, it may not be the most efficient solution. This is especially true if you already have investment in storage array technologies.

Integrating SRM with Storage Array–Based Replication

VMware Site Recovery Manager (SRM) can be, and frequently is, integrated with array-based replication technologies, yielding an enterprise-class business continuity solution. This kind of configuration allows for replication and site recovery via common VMware tools but with the power of rock-solid, array-based replication engines that offload all the replication overhead from the hypervisor.

Only supported storage arrays work with VMware SRM via a vCenter Storage Replication Adapter (SRA) plug-in that allows vCenter and the storage array to understand and communicate with each other.

Array-Based Replication

When it comes to replication technologies, array-based replication technologies sit at the bottom of the stack, below hypervisors, logical volume managers, and applications.

In array-based replication technologies, the storage array does all the hard work of replication, freeing up host resources to serve the application. However, vanilla array-based replication technologies—those that don't have special software to make them application aware—do not understand applications. So we consider array-based replication to be unintelligent, or more important, the replica volumes it creates will usually be *crash consistent* at best. This means that the state of the application data in the remote replica volume will not be ideal for the application to recover from, and additional work may be required in order to recover the application from the replica data. This is one of the major reasons that array-based replication is losing popularity and is often being replaced by application-based replication technologies.

NOTE It is possible to integrate array-based replication technologies with applications. An example is the synchronous replication API that Microsoft made available with Exchange Server 2010 and 2013 that allowed storage array vendors to integrate their replication technologies with Exchange Server, effectively replacing the native Exchange Server log-shipping application-layer replication engine with the synchronous block-based replication engine in the storage array. That provided the best of both worlds: application awareness as well as offloading of the replication overhead to the storage array.

That said, array-based replication is still solid and extremely popular, and it can be integrated with applications and hypervisors via APIs.

Storage arrays tend to implement asynchronous replication in one of two ways:

- Snapshot based
- Journal based

Let's take a look at both.

Asynchronous Snapshot-Based Replication

Many storage arrays implement asynchronous replication on top of snapshot technology. In this kind of setup, the source array takes periodic snapshots of a source volume—according to a schedule that you configure—and then sends the data in the snapshot across the replication network to the target array, where it's then applied to the replica volume. Obviously, in order to do this, an initial copy of the source volume needs to be replicated to the target system, and from that point on, snapshots of the source volume are then shipped to the target system.

Because *you* configure the schedule, you have the ability to match it to your SLAs and RPOs. So, for example, if you have agreed to an RPO of 10 minutes for a particular volume, you could configure the replication schedule for that volume to take a snapshot every 5 minutes and replicate the snapshot data to the target volume also every 5 minutes. This would give plenty of time for the snapshot data to be transmitted over the replication network and applied to the replica volume and keep you comfortably within the RPO of 10 minutes for that particular SLA.

It's important to note that configuring a replication interval of 10 minutes for a volume that you have agreed a 10-minute RPO for, will not be good enough to achieve the 10-minute RPO. After all, the data contained within the snapshot has to be sent across the replication network and then applied at the remote site, all of which takes additional time. Because of this, some storage arrays allow you to specify an RPO for a volume, and the array then goes away and configures an appropriate replication interval so that the RPO can safely be achieved. This approach also opens up the potential for the array to prioritize the replication of volumes that are approaching their RPO threshold over those that are not.

WARNING

If your array's asynchronous replication engine is based on snapshot technology, it's also important to understand whether the snapshots used for replication will count against the maximum number of snapshots supported on the system. For example, if your array supports a maximum of 256 snapshots, and you have 32 replicated volumes, each of which maintains two concurrent snapshots as part of the replication setup, those 64 replication-based snapshots could count against your system's 256 snapshots. Be careful to check this out when configuring a solution based on snapshot-driven replication. This is covered further in the discussion of array-based snapshots toward the end of the chapter.

WARNING

When working with asynchronous replication solutions that are built on top of snapshot technology, make sure that you understand the underlying snapshot *extent size*, as it can have a significant impact on network bandwidth utilization. For example, a large extent size, say 1 MB, will have to replicate 1 MB of data over the replication network every time a single byte within that extent is updated. On the other hand, a smaller extent size, such as 64 KB, will have to replicate only 64 KB over the replication network each time a single byte in an extent is updated. So smaller is better when it comes to replication snapshot extent sizes.

A good thing about snapshot-based replication is that as long as your array has decent snapshot technology, it will coalesce writes, meaning that if your app has updated the same block of data a thousand times since the last replication interval, only the most recent contents of that data block will be shipped over the replication network with the next set of deltas (a term used to refer to changed blocks), rather than all one thousand updates.

Asynchronous Journal-Based Replication

The other popular method for implementing array-based asynchronous replication is via journal volumes. Journal-based asynchronous replication technologies buffer new writes to either cache or dedicated volumes known as either *journal volumes* or *write intent logs* before asynchronously shipping them across the replication network to the target array. Let's take a closer look at how it usually works under the hood.

In a journal-based asynchronous replication configuration, as new write data arrives at the array, it follows the normal procedure of being protected to mirrored cache and then being signaled back to the host/app as complete. However, as part of this process, the write I/O is tagged as being part of a replicated volume to ensure that it gets copied to the journal volumes as well as being copied over the replication network to the remote target array when the next replication interval arrives. The gap between the replication intervals depends entirely on your array-replication technology and how you've configured it. However, it's usually every few seconds, meaning that the kinds of RPOs offered by journal-based asynchronous replication solutions are usually lower than those offered by snapshot-based replication configurations—which tend to offer minimum RPOs of 5 minutes.

Journal-based asynchronous replication can apply write sequencing metadata so that when the contents of the journal volume are copied to the remote system, writes are committed to the target volume in the same order that they were committed to the source volume. This helps maintain consistency of the remote replica volumes. However, if, for whatever reason the system can no longer write updates to the journal volumes (let's say the journal volumes have filled up or failed), the system will revert to maintaining a bitmap of changed blocks. This mode of operation, known as *bitmapping*, allows you to keep track of changed blocks in your source volume so that only changes are sent to the remote array, but it doesn't allow you to maintain write sequencing.

NOTE Some arrays implement asynchronous replication journaling in memory, in a construct often referred to as either a *memory transit buffer* or *sidefile*. Obviously, the amount of memory available for replication journaling is limited, and during extended periods of replication network downtime, these systems usually fall back to either using journal volumes, or switching to bitmapping mode where write ordering is not maintained.

Implementations differ as to whether the source array pushes updates to the target array or whether the target array pulls the updates. For your day-to-day job, things like this should not concern you, certainly not as much as making sure you meet your SLAs and RPOs.

So that's how journal-based asynchronous replication works. Let's look at some of the important points.

It's obviously vital that these journal volumes are big enough and performant enough to cater for buffering your replication traffic. On the capacity front, this is especially important when the network link or remote array is down for any sustained period of time. As with all sizing exercises, it's a trade-off: if you size the journal volumes too big, then you're underutilizing resources and wasting money, but if you size them too small, when the replication network or the remote array is down, they can fill up and cause your replicated volumes to require a full sync when the problem is fixed. A full sync is not something you want to be performing often.

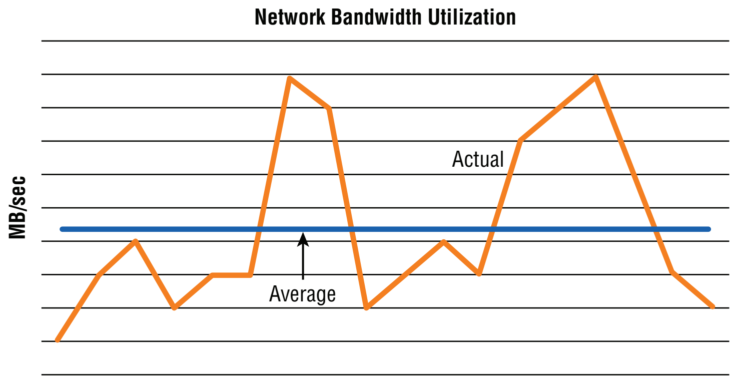
As an example, assume a 2 TB volume is being replicated. When the replication network goes down—let's say a maintenance contractor hit a cable when digging up the road outside your data center—all changes to the source volume will be buffered to the journal volumes until the replication network comes back up. If the replication link is fixed before the journal volumes fill up, all you have to replicate to the target array are the changes contained in the journal volumes. However, if the journal volumes filled up during the network link outage, your replicated volumes would almost certainly be marked as failed and require you to perform a full sync—copying the entire 2 TB volume across the replication network to the remote array. This is likely to be a lot more than just the changes contained in the journal volumes. Even over 8 Gbps of bandwidth between two sites that are less than 100 miles apart, copying terabytes of data can take hours.

WARNING

Asynchronous replication solutions allow you a lot more leeway when speccing replication network connections. For starters, they don't have to be sized to cater for peak bandwidth. Arguably, they don't need to be as reliable as they do for synchronous replication. However, be careful not to under-spec them. If the bandwidth is too low or they are too unreliable, you could find yourself struggling to operate consistently within your agreed-upon SLAs.

Figure 8.6 shows network bandwidth requirements for asynchronous replication. The bandwidth utilization is captured at an hourly interval, with an average line included.

FIGURE 8.6 Bandwidth requirements for asynchronous replication



Replication Topologies

The replication topologies covered here are not restricted to array-based replication. However, this section focuses on how different replication topologies are implemented by using array-based replication, because the most complex replication topologies tend to be implemented around array-based replication.

Different storage arrays support different replication topologies. Not every array supports every replication topology, though. Make sure you do your research before trying to implement a complex replication topology, as you may find yourself in an unsupported situation. These different topologies are often a mix of synchronous and asynchronous replication, with two, three, or even four arrays, located in multiple sites at various distances apart.

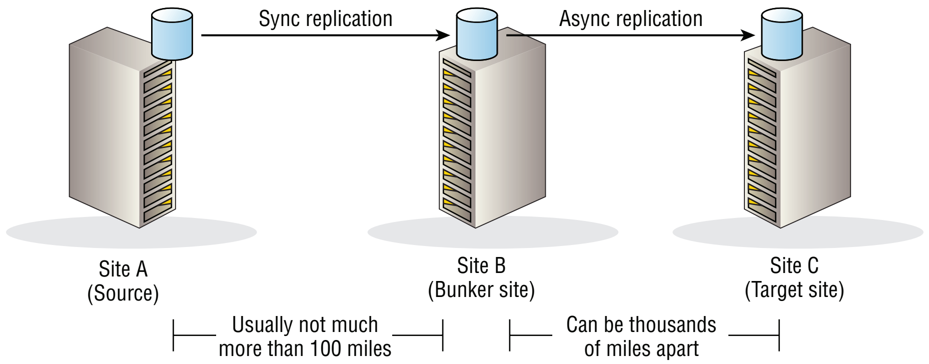
Let's take a look at some example replication topologies.

NOTE The replication topologies covered throughout this section are by no means exhaustive. Plenty of other configurations are possible. Most of them tend to be complicated to configure and maintain.

Three-Site Cascade

As the name suggests, Three-Site Cascade is based around three sites: a source site, a target site, and a bunker site between them. This configuration is shown in Figure 8.7.

FIGURE 8.7 Three-Site Cascade topology



Arrays at the source and target site can't communicate with each other directly. Any data that you want to replicate from an array at the source site to an array at the target site has to first go to an array in the bunker site. A common example is a source and bunker site that are less than 100 miles apart and configured with synchronous replication. The target site is then over 100 miles away and replicated to asynchronously.

Because the target site is usually so far away from the source and bunker sites, this kind of topology lends itself well to high-performance requirements with low RPOs but the ability to survive large-scale local disasters. Take a minute to let that settle in. First of all, this solution can survive large-scale local disasters because the target site can be thousands of miles away from the source site. It can also be high performance, because the synchronous replication between volumes in the source and bunker sites can be close enough to each other that the network latency imposed on the replication traffic is low. And RPOs can be low (in fact, they can be zero), because synchronous replication offers zero data loss. Of course, if you do experience a large-scale local disaster and lose both your source and bunker sites, you'll have to recover applications at the target site that is asynchronously replicated to, so you will be lagging behind your source site and you will stand a good chance of losing some data.

Three-Site Cascade allows your applications to remain protected even if you lose the source site. This is because the bunker and target sites still have replication links between them, meaning that replication between these two sites can still occur. In simple two-site replication models, if you lose your source site and failover applications to the target site, your target site is "on its own" until you recover the source site and reconfigure replication back from the target to the original source site.

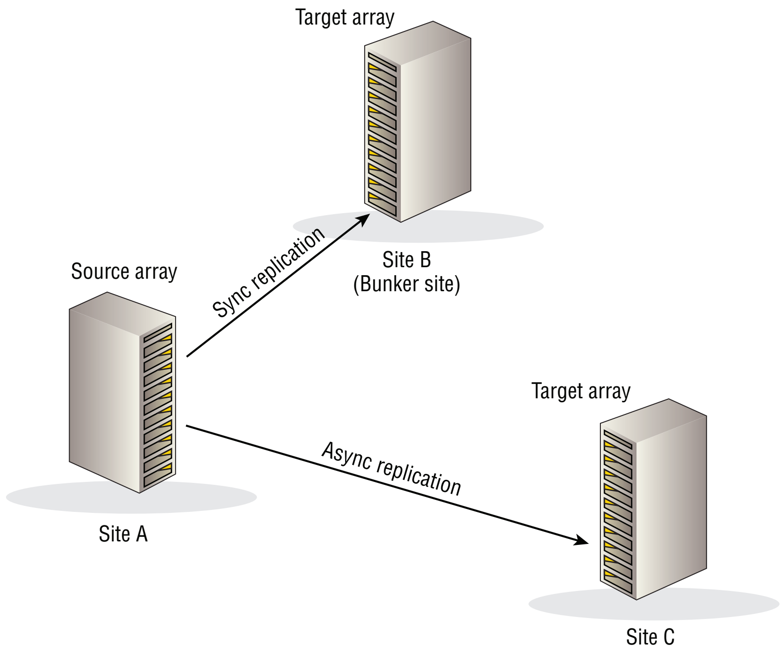
At least one major weakness of the Three-Site Cascade model is that the loss of the bunker site severs the link between the source and target site. Data at the source site cannot be replicated to the target site, and the target site starts to lag further and further behind.

Three-Site Multi-Target

Three-Site Multi-Target has the source array simultaneously replicate to two target arrays. This topology is potentially more resilient than Three-Site Cascade, as the bunker site is no longer a soft spot in the solution. The source array has visibility of both of the other arrays in the configuration and is configured to replicate to them both, without having to rely on arrays in the bunker site. However, the major downside to Three-Site Multi-Target is that there usually isn't a connection between the two target arrays. If the source site is lost, the surviving two arrays cannot replicate to each other.

As shown in Figure 8.8, in the Three-Site Multi-Target topology, one of the remote sites is usually within synchronous replication distance, and the other a lot farther away. This kind of setup allows the topology to be able to provide zero data loss (via the local synchronous replication link) as well as the ability to ride out large-scale local disasters thanks to the asynchronous replication to the array that's a lot farther away.

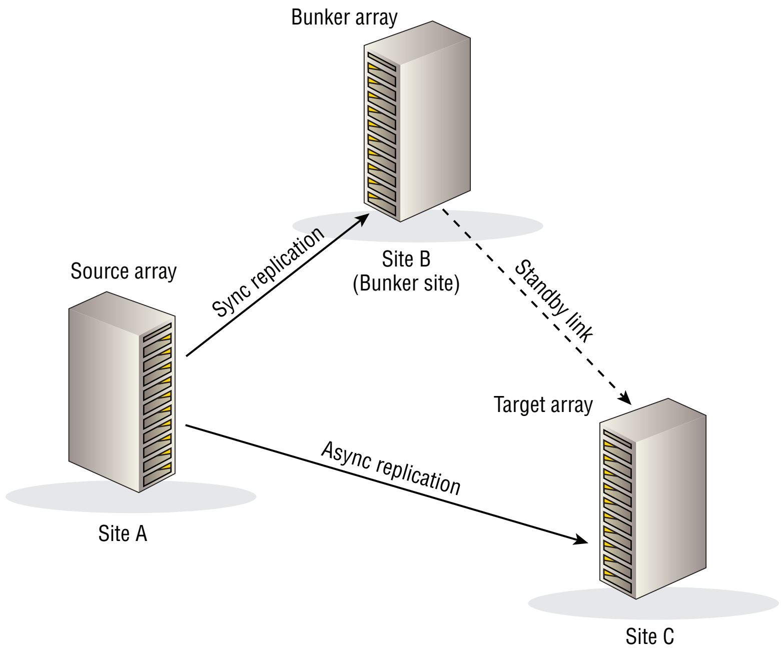
FIGURE 8.8 Three-Site Multi-Target topology



Three-Site Triangle

As shown in Figure 8.9, the Three-Site Triangle topology is almost exactly the same as Three-Site Multi-Target. The only difference is the existence of an additional replication link between the two target arrays in site B and site C. As indicated in Figure 8.9, this additional replication link between the two target arrays is usually in standby mode, meaning that under normal operating circumstances, it's not used to send data. However, it can be used to replicate data in the event that the source site or source array is lost. This helps remove the major weakness of Three-Site Multi-Target, where if the source array is lost, although you have two surviving arrays, there is no way for them to replicate with each other, and you're left with two copies of your data but no way to configure the two surviving arrays into a disaster-resilient pair.

FIGURE 8.9 Three-Site Triangle replication topology



As long as your array's replication technology supports it, it's entirely possible that in the event of a disaster that causes you to lose your source array, your two remaining target arrays could communicate with each other, determine the differences between the copies of data they hold, and have to ship only the difference to each other, rather than requiring a full sync.

Local Snapshots

First, let's clarify the terminology. *Snapshots* are local copies (replicas) of data. *Local* means the snapshot exists on the same host or storage array as the source volume, making snapshots useless as recovery vehicles if you lose access to your source array. If you lose access to your source array, you obviously lose access to the source volumes *and* the snapshots.

Snapshots are what we call *point-in-time (PIT)* copies of system volumes. They provide an image of how a volume or system was at a particular point in time—at 5 p.m., for example. These snapshots can be used for a lot of things, such as rollback points, backup sources, scratch volumes to perform tests against, and so on. Snapshots are pretty flexible and are widely used in the real world.

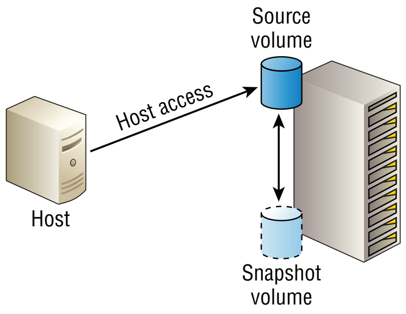
Anyway, back to the terminology for a moment. Let's define exactly what we mean when we use the terms *point-in-time* and *local*.

Local refers to the created copies existing on the same system as the source volumes. If they are host-based snapshots, the snapshots exist on the same host as the source volume. If they are array-based snapshots, the snapshots exist on the same array as the source volumes.

Point-in-time snapshots are copies of a volume as it is at a certain time. For example, a PIT snapshot might be a copy of a volume exactly as that volume was at 9 a.m. This means they are not kept in sync with the source. This is the exact opposite of remote replicas, which are usually kept in sync, or semi-sync, with the source volume.

The terms *source volume* and *primary volume* refer to the live production volume, and the terms *source array* and *primary array* refer to the live production array hosting the source volume. Figure 8.10 shows a simple snapshot configuration with some terminology mapped to the diagram.

FIGURE 8.10 Local snapshot



Depending on the snapshot technology you're using, you'll be able to create read-only snapshots, read-write snapshots, or both.

WARNING

A vital point to note about snapshots is that they are not backups! Make sure you never forget that. Snapshots exist on the same system as the source's volume, and if you lose the source volume or source system, you also lose your snapshots. However, if you're replicating data to a remote target array, and then you take snapshots of the remote target volumes, these can work as backup copies of your source volumes because they're on a separate system, potentially in a separate site. If you lose your source system or source site, you will still have access to your remote snapshots.

Hypervisor-Based Snapshots

Most modern hypervisors now provide native snapshotting technologies. As with hypervisor- and host-based replication technologies, these snapshots consume host-based resources and do not benefit from offloading the snapshot process to the storage array. That said, they are improving and are gaining in popularity.

As VMware is a major player in the hypervisor space and has a relatively mature integrated snapshot stack, let's take a closer look at how virtual machine snapshots work in a VMware environment.

VMware Snapshots

VMware vSphere offers native virtual machine (VM) snapshotting capabilities as well as an API that third-party vendors can utilize to create and manipulate VM snapshots.

VM snapshots are PIT snapshots that preserve the state and the data in a VM at the time at which the snap was taken. For example, if you take a snapshot of a VM at 5 p.m., you have preserved an image of how that VM was at precisely 5 p.m. This snap includes the state of the VM, including whether it was powered on at the time the snap was taken, as well as the contents of the VM's memory.

When creating VM snapshots, there is an option to quiesce the VM. *Quiescing* a VM is all about making sure the snap you take is consistent.

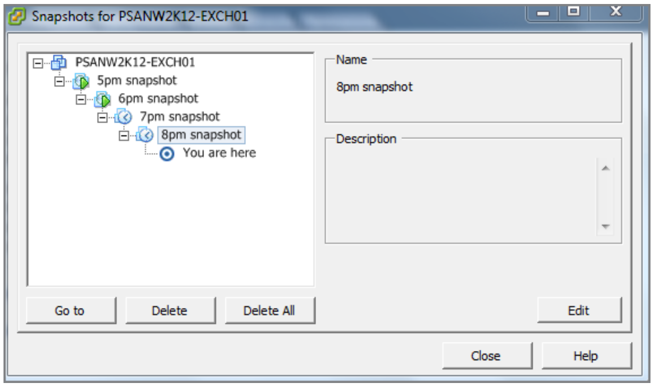
For Microsoft servers—as long as VMware Tools is installed on the VM being snapped—a call is made to the VSS to quiesce the VM. VSS ensures that all filesystem buffers are flushed, and new I/O to the filesystem is held off until the snap completes. It is also possible to quiesce any VSS-aware applications such as Microsoft Exchange Server, Microsoft SQL Server, Oracle Database, Microsoft SharePoint Server, and Microsoft Active Directory. This ensures not only filesystem-consistent snapshots, but also application-consistent snapshots.

If VSS cannot be used to quiesce a system (for example, on non-Microsoft operating systems), the VMware proprietary SYNC driver manages the quiescing of the system. This can often be filesystem quiescing only. The SYNC driver effectively holds incoming writes on a system as well as flushes dirty data to disk in an attempt to make a filesystem consistent.

It is possible to have multiple snapshots of a VM. These snapshots form a tree, or chain, of snapshots. You can delete any snapshot in the tree and revert to any snapshot in the tree. However, having a large or complicated snapshot tree structure carries a performance overhead. So does having lots of snaps of a VM that experiences a high rate of change. So be careful when creating lots of snapshots. Just because you can doesn't mean you should.

Figure 8.11 shows a snapshot chain.

FIGURE 8.11 VMware Snapshot Manager snapshot chain



NOTE

Take another look at Figure 8.11, which shows a snapshot chain in VMware Snapshot Manager. Notice the *You are here* at the very bottom of the snapshot chain. By default, VMware snapshots work so that you are working off the last snapshot in the chain. If you delete your snapshot, its contents will need to be merged in a process VMware calls *consolidation.* Assume you have a single parent (source) VM and a single snapshot. Your VM is running off the snapshot, and unless you want to lose all changes you have made to the VM since the snapshot was taken, when deleting the snapshot you need to merge the contents of the snapshot with the parent volume. This rolls up all changes since the snapshot was created back to the parent volume. If you don't consolidate the contents, you will lose all changes made since the snapshot was created.

In a VMware environment, when a snap of a VM is created, each VMDK associated with the snapped VM gets a delta-VMDK file that is effectively a redo file. There are also one or two metadata files created for storing the contents of the VM's memory, as well as snapshot management. All of these delta files are *sparse files*—space efficient—and work like traditional copy-on-write snapshots that start filling up with data as you make changes. Delta files obviously consume space relative to the change rate of the VM and can grow to a maximum size equal to that of the parent disk.

WARNING

Snapshots are not backups! However, they are often used by VM backup applications, such as Veeam Backup & Replication. When making a backup of a VM, backup applications make a call to the VMware snapshot API to create a snap of a VM to be used as a consistent PIT copy that can be used for creating the backup. In these situations, once the backup is complete, the snapshot of the VM is released. In addition (and this is configurable and also depends on your version of vSphere) VM snapshots by default are created in the same location as the source VM that is being snapped. This usually means they have the same datastore, and therefore the same physical drives on the storage array. So a snapshot is definitely no use to you as a backup if your storage array or underlying RAID groups get toasted!

As I've mentioned backing up Microsoft servers and Microsoft applications, it is only fair to point out that the Microsoft Hyper-V hypervisor also offers fully consistent application backups for all applications that support Microsoft VSS: Exchange Server, SQL Server, Oracle Database on Windows, SharePoint Server, Active Directory, and so on.

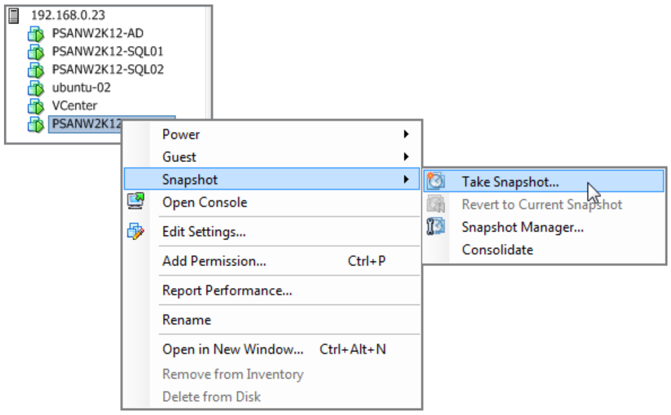
Now let's take a quick look at how to take a VM snapshot. Exercise 8.2 shows how to do this using the vSphere Client.

EXERCISE 8.2

Taking a Manual VM Snapshot

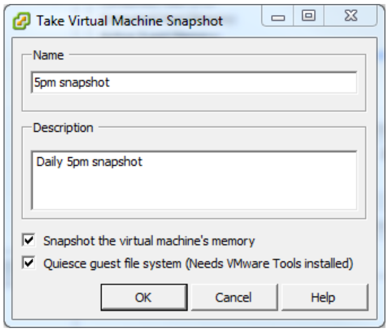
In this exercise, you will walk through taking a manual VM snapshot by using the vSphere Client.

1. From within the vSphere Client, locate the VM that you wish to take a snapshot of and right-click.
2. Choose Snapshot ➪ Take Snapshot.



EXERCISE 8.2 ***(continued)***

3. On the screen that appears, give the snapshot a name and an optional description. Make sure to select the two check boxes, Snapshot The Virtual Machine's Memory and Quiesce Guest File System (Needs VMware Tools Installed).


Take Virtual Machine Snapshot

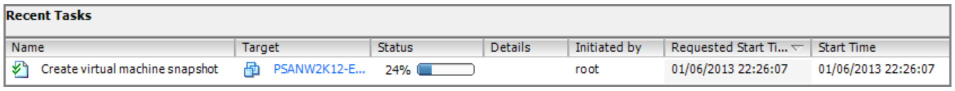
Name: 5pm snapshot

Description: Daily 5pm snapshot

☑ Snapshot the virtual machine's memory
☑ Quiesce guest file system (Needs VMware Tools installed)

4. Click OK.

5. The Recent Tasks view at the bottom of the vSphere Client screen shows the progress of the snapshot operation.


Recent Tasks

Name	Target	Status	Details	Initiated by	Requested Start Ti...	Start Time
Create virtual machine snapshot	PSANW2K12-E...	24%		root	01/06/2013 22:26:07	01/06/2013 22:26:07

Array-Based Snapshots

As with array-based replication, array-based snapshots offer the benefit of offloading the snapshot *work* from the host system to the array. This can free up CPU and memory on the host, as well as provide faster snapshots and potentially quicker recoveries from snapshots. However, it can also add complexity to the configuration. As with all things, when possible, you should try before you buy.

As with hypervisor-based snapshots, all good array-based snapshot technologies can be made application aware by integrating with products such as Microsoft VSS to provide application-consistent snapshots. Also, when augmented with backup software, the combination of the backup software and the array-based snapshots can inform applications that they have been backed up, allowing them to truncate log files and do other tidy-up exercises often associated with backups.

In the next few pages, you will look closely at array-based snapshot technologies and concentrate on how they are implemented, as well as the pros and cons of each.

Two primary types of array-based snapshot technologies are commonly used:

- Space-efficient snapshots
- Full clones

These technologies sometimes go by various names, so I will try to be relatively consistent in this chapter. Sometimes I might call *space-efficient snapshots* just *snapshots*, and sometimes I will call *full clones* just *clones*. Also, while on the topic of vernacular, I will use the term *source volume* to refer to the original volume that the snapshot is taken from. I will also draw snapshots with dotted lines in the figures, to indicate that they are space efficient.

Space-Efficient Snapshots

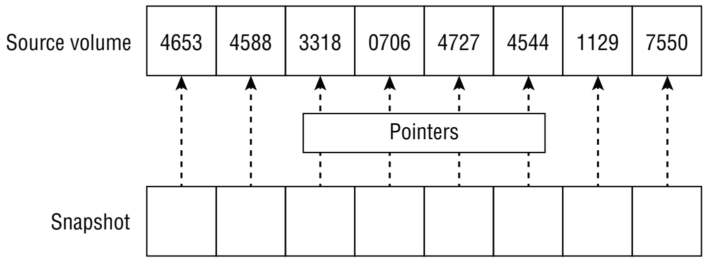
Space-efficient snapshots are *pointer based*. There are two significant concepts to consider when dealing with such snapshots:

Space efficient tells us that the snapshot contains only changes made to the source volume since the snapshot was taken. These changes are sometimes referred to as *deltas*.

Pointer based tells us that when the snapshot is created, the entire address space of the snapshot is made up of pointers that refer I/O requests back to the source volume.

Figure 8.12 shows the concept of pointers.

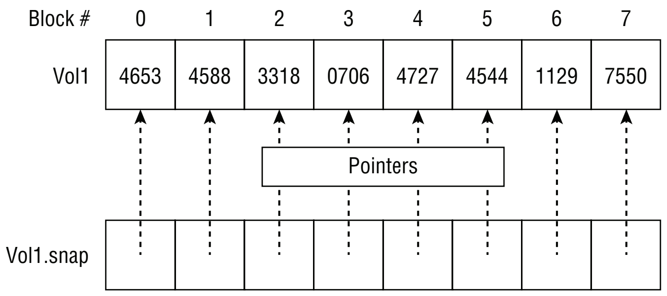
FIGURE 8.12 The use of pointers in snapshots



Let's look at a quick, simple example. Say that we have a source volume called `Vol1`. `Vol1` has eight data blocks numbered 0–7. At 1 p.m. on Monday, we make a space-efficient snapshot of `Vol1`. At that time, all blocks in `Vol1` are as shown in Figure 8.13. The snapshot of `Vol1` is called `Vol1.snap`. When we create the snapshot, `Vol1.snap` consumes *no* space on disk. If a host mounts this snapshot and reads its contents, all read I/O will be redirected back to the original data contained in `Vol1`. `Vol1.snap` is basically an empty volume that contains only pointers (usually in memory) that redirect read I/O back to the primary volume containing the data.

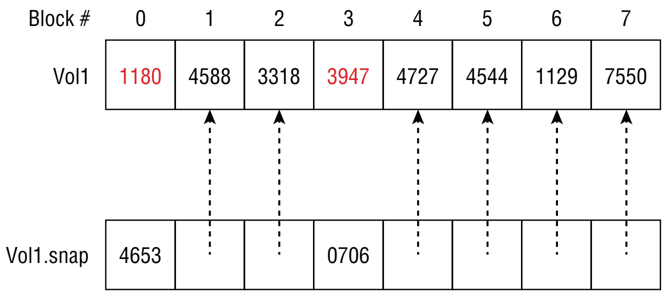
Next let's assume that by 2 p.m. `Vol1` has made updates to blocks 0 and 3. Now if a host wants to read data in `Vol1.snap`, it will be redirected back to the source volume (`Vol1`) *except when reading blocks 0 and 3*. Because blocks 0 and 3 have changed since the time the snapshot was created, the original contents (4653 and 0706) have been copied to `Vol1.snap` in order to preserve an image of `Vol1` as it was at 1 p.m.

FIGURE 8.13 Source and snapshot volume at 1 p.m.



At 2 p.m., the snapshot configuration looks like Figure 8.14, and the snapshot volume still represents an exact image of how the primary volume was at 1 p.m.

FIGURE 8.14 Source and snapshot volumes at 2 p.m.



Snapshots are also instantly available, which means that as soon as you click the Create button, figuratively speaking, the snapshot is available to be mounted and used.

Snapshots can also be read/write, making them flexible and opening them up to lots of use cases. However, if you mount a snapshot as read/write and issue writes to it, it will obviously no longer represent the state of the source volume at the time the snap was taken. This is fine, as long as you do not plan to use the snapshot to roll back to the point in time at which the snap was taken!

If you're new to snapshot technology and this is still unclear, go grab a coffee, remove any distractions, and reread the above. It is vital that you understand how space-efficient snapshots work.

Full-Clone Snapshots

Full-clone snapshots, sometimes called just *full clones* or *clones*, are not space efficient. They don't use pointers, so they can be quite a lot simpler to understand. However, they are still PIT copies of source volumes.

At the time a clone is created, an exact block-for-block copy of the source volume is created. If the source volume is 2 TB, the clone will also be 2 TB, which will consume 4 TB of storage. You can see how full clones can quickly eat through storage.

As with snapshots, clones can be read/write, and as soon as you write to a clone (or snapshot), it no longer represents the state of the source volume at the time the clone was created. This makes it no use as a vehicle to roll back to the state the source volume was in at the time the clone was created. However, it can be useful for testing and other purposes.

How Instantaneous Cloning Works

Most array-based cloning technologies provide instantaneous clones. Just the same as with space-efficient snapshots, this means that they are immediately available after you click Create.

But how can a clone of a large multiterabyte volume be instantly available? Surely it takes a long time to copy 2 TB of data to the *clone* volume?

Instantaneous clones generally work as follows. When the clone is created, the array starts copying data from the source volume to the clone across the backend as fast as possible, although it is careful not to slow down user I/O on the front end. If you read or write to an area of the clone that is already copied, obviously that is fine. However, if you want to read an area of the clone that has not been copied yet, the array will either redirect the read I/O to the source volume for the blocks in question or it will immediately copy those blocks across from the source to the clone so that the read I/O does not fail. Either way, the operation is performed by the array in the background, and hosts and applications should not notice any difference.

If your clones are read/write clones, any write I/Os to an area of a clone that has not yet been copied from the source can just be written straight to the clone volume, and the respective blocks from the source volume no longer need to be copied across.

Figure 8.15 compares a space-efficient snapshot of a 2 TB volume with a full clone of a 2 TB volume at the time the snap/clone was taken.

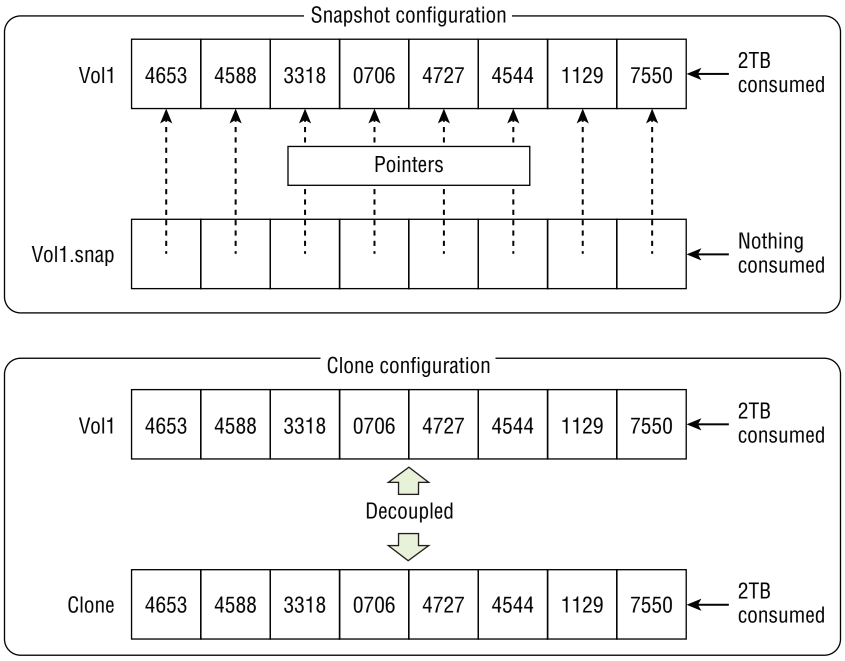
Obviously, over time the snapshot starts to consume space. How much space depends entirely on your I/O profile. Lots of writes and updates to the source volume will trigger lots of space consumption in your snapshot volume.

So if full clones consume loads of disk space, why bother with them, and why not always go for space-efficient snapshots? There are a couple of common reasons:

- Reading and writing to a full clone has no effect on the performance of the source volume. You can hammer away at the clone without affecting the source volume's performance. This can be useful for a busy database that needs backing up while it is busy; you can take a clone and back up the clone. You could also use a clone of the same busy database as a reporting database so that reports you run against the clone don't impact the performance of the database. This works because the full clone is fully independent of the source, should be created on separate spindles, and contains no pointers. Conversely, space-efficient snapshots share a lot of data with their source volumes. If you hammer the snapshot with intensive read and write I/O, you will reference some data blocks shared with the source and therefore hammer the source volume.

- Full clones are fully independent of the source volumes. If the physical drives that the source volume is on fail, the clone will not be affected (as long as you don't keep the source and clone on the same drives). Because space-efficient snapshots rely on the source volume, if the disks behind the primary volume fail, the source and the space-efficient snapshot will fail.

FIGURE 8.15 Comparing the space consumption of a space-efficient snapshot and a full clone



In the modern IT world, where budgets are tight and ever shrinking, snapshots tend to be far more popular than full clones, so we'll spend a bit more time going into the details of array-based, space-efficient snapshots.

Snapshot Extent Size

Snapshots by their very nature are space efficient. One of their major strengths is that they take up very little space. So when it comes to space-efficient snaps, the more space we can save, the better. Smaller snapshot extent sizes are better than large snapshot extent sizes. So a snapshot extent size of 16 KB is better than one of 256 KB. Although these might seem like small numbers, on large systems that have hundreds of snapshots, the numbers can quickly add up.

The same goes for asynchronous replication technologies that rely on an array's underlying snapshot technology. In these situations, arrays with larger snapshot extent sizes will have to replicate larger amounts of data to the remote system.

Reserving Snapshot Space

Some storage arrays, especially older arrays, require you to set aside capacity to be used as snapshot reserve space. This space is then used to store data associated with snapshots. If you expect your source volume to change a lot, you'll have to reserve quite a lot of space for snapshot data, whereas if your source volumes won't change a lot, you won't have to reserve as much space for snapshot data. The problem with this approach is twofold:

- It's often difficult to know if your source volumes will be subject to high or low rates of change.
- Even if you expect your source volumes to experience high rates of change, exactly how much space to set aside is still almost impossible to know.

This often results in snapshot reserve space being oversized, meaning that you have capacity sitting in your array that is ring-fenced for snapshot data but is potentially never used. That's not great.

Fortunately, most modern storage arrays don't work this way. Most dynamically allocate space to snapshots on an as-needed basis, with the space coming from the general pool of storage that all volumes have access to (rather than a dedicated ring-fenced pool of storage). In these more modern architectures, it is still possible to ensure that snapshot space is never taken from the same pool as space for the source volume—sometimes seen as good practice for both availability and performance reasons—by having multiple pools. The important point is that both of these pools can be used for source volumes and snapshot data; you may just choose not to have space for the source and associated snapshots from the same pools. For example, pool A might be used for source volume 1, and pool B for snapshots of source volume 1. Pool B might be used for source volume 2, and pool A for snapshots of source volume 2. In this configuration, both pools are used for both source volumes and snapshots.

This sharing of pools by both source volumes and snapshot data can cause problems if space is getting low, and most arrays will allow you to configure policies that dictate what happens to space and snapshots when capacity gets low. For example, when capacity reaches 90 percent, you might stop any new snapshots from being created, and when capacity hits 95 percent, you might start deleting the oldest snapshots to free up data for the growth of source volumes.

Snapshots and Tiering

Because snapshots are often considered less important than their associated production source volumes, it can be useful to store them on lower-tier storage. Most storage arrays will allow you to specify that snapshot volumes be created from lower tiers of storage.

However, a lot of modern arrays allow snapshots to be managed by the array's auto-tiering algorithm. This allows snapshots to be created in the higher tiers, and then if not used, trickle down, over time, to the lower tiers. This can be useful in situations where your snapshot is accessed in the first few days after creation but then not accessed for the next few weeks before eventually being deleted. If controlled by the array auto-tiering algorithm, a snapshot can start its life on high-performance storage and stay there while it's being used, but then spend its last few days of life on slower, cheaper storage while it isn't being accessed.

Crash-Consistent and Application-Aware Snapshots

As is the case with standard array-based replication, array-based snapshot technologies in their out-of-the-box condition are not application aware, meaning that the data inside the snapshot will be what we term *crash consistent*.

NOTE The term *crash consistent* indicates that the data will be only as consistent as if the server had unexpectedly crashed.

The reason that snapshots, without any special software to integrate them with applications, are crash consistent is that most applications and databases cache data in local buffers on the server, and lazily synchronize these updates to disk. At any point in time, the data on disk does not accurately represent the current state of the application. This is extremely important to understand. Think of it like sending an email home, saying you will arrive at the train station tomorrow at 10 a.m., only then to write another email saying you've changed your plans and will now fly back and be at the airport at 9 a.m. If your second email never arrives, you won't be picked up at the airport, and your ride home will have a wasted trip to the train station. It's the same with applications writing to disk: the data on disk might not be the latest copy of the state of the application, so snapshotting the contents of the disk is no use unless you first flush the application's state to disk, which is part of the process known as quiescing.

Fortunately, most array-based snapshots can be endowed with application intelligence and therefore can be capable of creating application-consistent snapshots. This is usually achieved via application frameworks or application integration software. For example, as already mentioned, Microsoft VSS is an API and framework that allows third-party snapshot technologies (including array snapshot technologies) to integrate with Microsoft applications and produce application-consistent snapshots.

If you plan on relying on your snapshots to recover your application, you had better plan on integrating your snapshot technologies with your applications so that you get application-consistent snapshots. If you don't, you run the risk of your snapshots not being able to recover your applications, or if they can, facing a lot more work to accomplish this.

Copy-on-Write Snapshots

Copy-on-write (CoW) snapshots are a form of space-efficient snapshot that are less frequently used these days. They maintain the contiguous layout of the primary volume, which is great, but they also unfortunately have the overhead of first-write performance. Consider both of these points a little more closely:

- After a snapshot has been taken, the first write to any block on the source volume requires the original contents of that block to be copied to the snapshot volume for preservation. Only after this is done can the write operation overwrite the original contents of the block on the source volume. This can have a performance overhead associated with it. This is less of an issue on an SSD-based array or SSD volume, because no positional latency is incurred by having to move the heads and so on. Also, many arrays can hide this performance hit by performing operations in cache and asynchronously updating the contents of disk, although this tends not to scale very well if you plan on keeping lots of snapshots.
- CoW snapshots preserve the original contiguous layout of the primary volume. This is discussed in more detail in the following section.

Redirect-on-Write Snapshots

Redirect-on-write (RoW) snapshots are sometimes referred to as *allocate-on-write snapshots*. They are very different from copy-on-write snapshots. When a volume that is protected by a snapshot receives a write I/O, the original data in the source volume is not changed or copied to the snapshot volume. The data is frozen and becomes part of the snapshot via a very fast metadata operation. The new write that came into the source volume is simply written to a new location on the array via a form of redirect operation, and the metadata table for the source volume is updated to reflect the new location.

Let's look at a quick and simple example. Say you have a volume called `Vol1` that comprises 100 contiguous blocks, 0–99. You snap `Vol1` at 1 p.m., and then at 2 p.m. update the contents of block 50. With redirect-on-write, the contents of block 50 will be frozen in their original condition and made part of the address space of the snapshot volume; let's call it `Vol1.snap`. The write data that was destined to block 50 is redirected and written somewhere else; let's say block 365. The metadata for `Vol1` is updated so that logical block 50 now maps to physical block 365 on the disk.

NOTE Metadata writes are so fast that they don't impact volume performance.

The downside to redirect-on-write snapshots is that they tend to fragment the address space of the source volume. In our simple example, `Vol1` originally had 100 contiguous blocks on disk, blocks 0–99. After updating the contents of block 50, `Vol1` has one block (1 percent of its contents) in a random location elsewhere in the system. This is shown in Table 8.1.

TABLE 8.1 Impact of redirect-on-write snapshot behavior on volume address space

Logical Address	Physical Address
0	0
1	1
2	2
3	3
...	...
48	48
49	49
50	365
51	51
52	52
...	...
97	97
98	98
99	99

This fragmentation of the source volume's address space can add up and eventually have a negative impact on volume performance. A volume that is heavily fragmented can no longer benefit from sequential reads and writes, forcing all access patterns to become random. This is much less of a problem on all-flash systems or systems with a lot of flash.

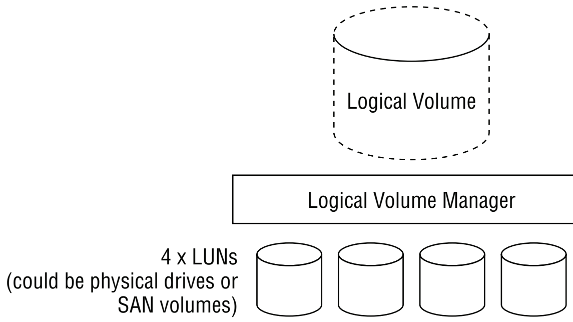
Host-Based Snapshots and Volume Managers

Another option for making and managing snapshots is via host-based volume managers. A *volume manager* is software that provides advanced host-based volume management functionality such as the following:

- Creating striped, mirrored, or RAID-protected volumes from multiple underlying physical LUNs. For example, it is still quite common for a host to be given multiple RAID-protected LUNs from a storage array, and then to have the host-based volume manager add a layer of striping on top of the volumes. This is shown in Figure 8.16.

- Dynamic adding and removing of LUNs as well as associated growing of volumes.
- Snapshot creation and management.

FIGURE 8.16 Volume manager creating a stripe on top of array-based LUNs



Like other types of snapshots, volume manager snapshots can be used by backup applications: a consistent snapshot of a volume is taken, and it is the consistent snapshot that is backed up. Regarding snapshot technologies, volume manger snapshots sit in the host-based camp, and as a result they consume host resources. They do not offload any of the legwork involved in snapshot management to the storage array.

On the plus side, they can provide filesystem-level consistency and can usually be configured to be read/write snapshots. However, both of these capabilities depend on the volume manager technology in question and sometimes on the Linux kernel in use. It's no more or less complicated than array-based snapshots. Whichever technology you choose, you need to do your research and testing before deploying in your live production environment.

Quite often, in order to make volume manager snapshots application consistent, you need to manually (probably via a script) quiesce your applications.

On the down side, volume manager snapshots often have an associated performance overhead. It is also quite common for volume manager snapshots to be visible and exportable only to the host that is running the volume manager.

They are, however, widely used in smaller shops and development environments.

Summary

In this chapter we discussed how replication technologies fit into the bigger picture of business continuity and disaster recovery, before diving deeper into the different types of replication common in today's storage world. We compared synchronous and asynchronous replication, and the impacts each has on recovery point objectives, application performance, network bandwidth requirement, and other factors. We discussed how replication technologies are implemented at different layers of the stack, including application, operating system, and storage array. We also discussed common replication topologies and how they protect businesses and their data from different types of disaster. We then finished the chapter discussing snapshot technologies and application consistency.

Chapter Essentials

Synchronous Replication Synchronous replication offers zero data loss, giving the best recovery point objective of all the different types of replication. However, it carries a performance penalty that increases as you add distance between your source and target volumes.

Asynchronous Replication Asynchronous replication practically guarantees to lose some of your data. However, it does not impose the performance penalty that synchronous replication does. Asynchronous replication allows you to have your source and target at almost limitless distances from each other and can usually be configured with a recovery point objective.

Application-Layer Replication Replication performed at the application layer levies the burden of replication on the host but offers excellent application awareness, ensuring that remote replicas are transactionally consistent.

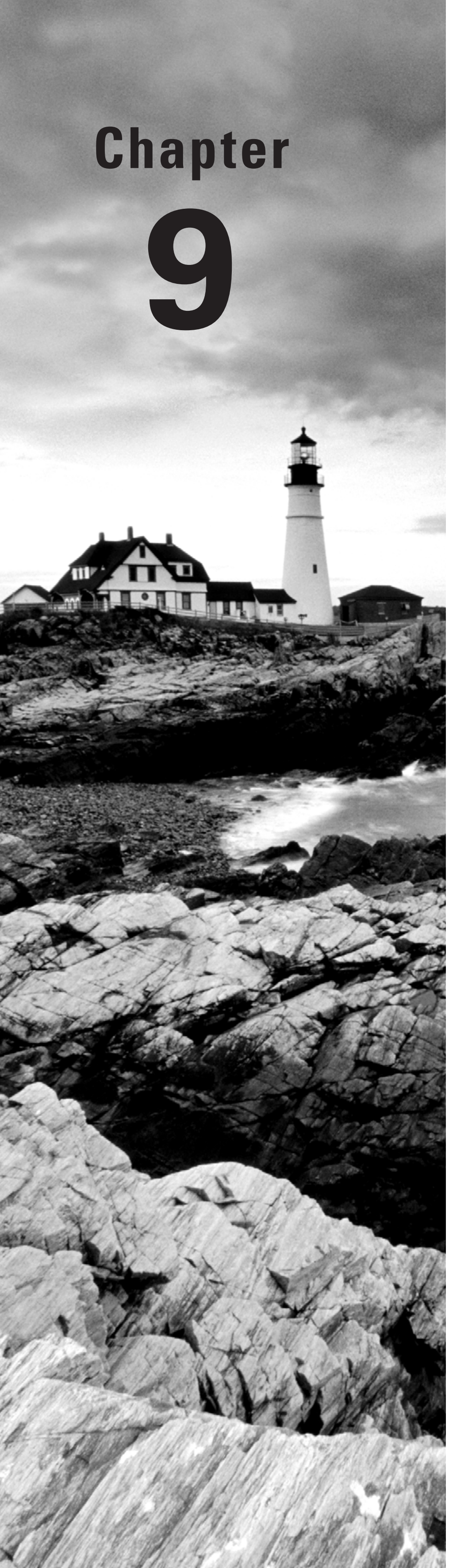
Space-Efficient Snapshots Space-efficient snapshots are point-in-time copies of volumes, or virtual machines, that provide an image of a volume, or virtual machine, at a specific time. These can be used as rollback points, sources for backups, or testing. They are absolutely not a replacement for backups!

Chapter
9

Storage Virtualization

TOPICS COVERED IN THIS CHAPTER:

- ✓ What storage virtualization is
- ✓ Host-based storage virtualization
- ✓ Network-based storage virtualization
- ✓ Controller-based storage virtualization
- ✓ In-band vs. out-of-band storage virtualization
- ✓ Use cases for storage virtualization
- ✓ How to configure controller-based virtualization
- ✓ Software-defined storage



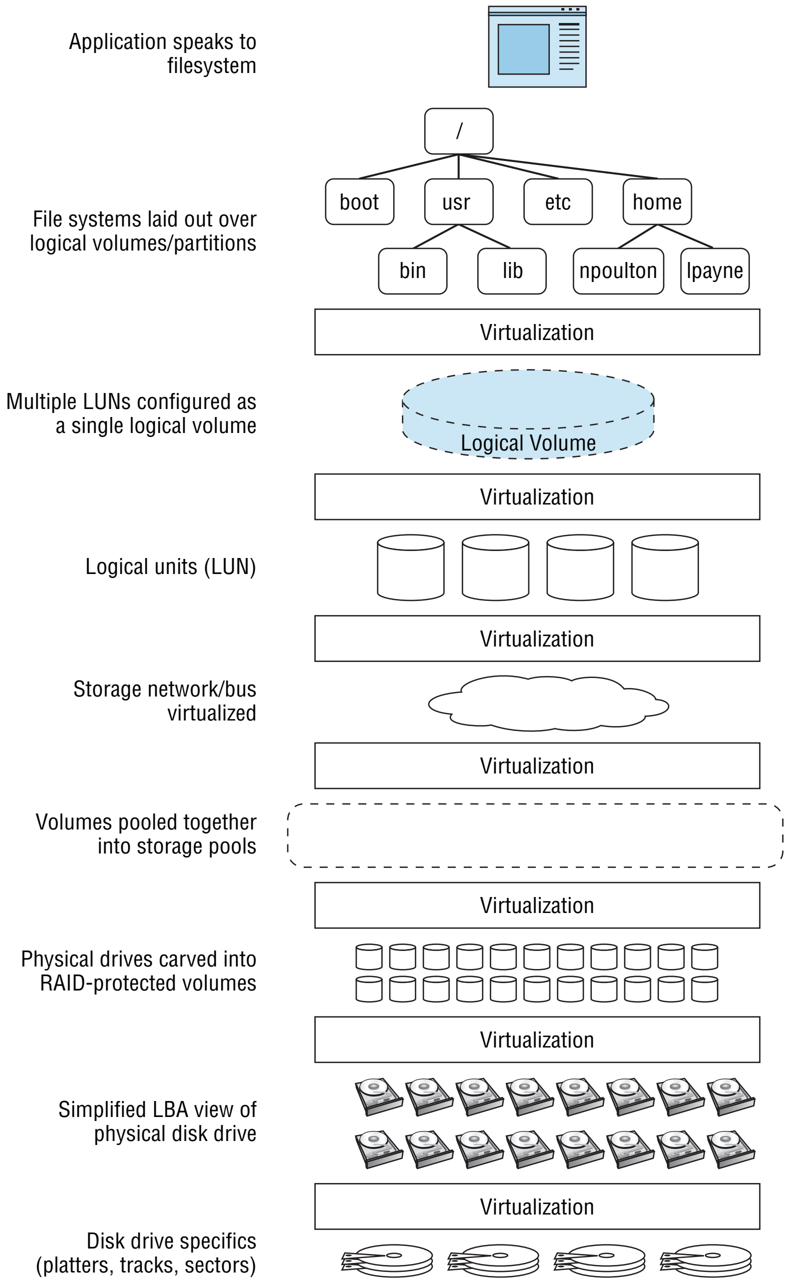
This chapter covers the popular forms of storage virtualization seen and implemented in the real world. We'll start out by explaining the many different layers of virtualization that exist in a typical storage stack, which have been there for years but are not often thought of as virtualization per se. We'll introduce the SNIA Shared Storage Model (SSM) and use this as a frame of reference for mapping different types of storage virtualization to where they fit in the SSM stack. We'll spend a fair amount of time talking about controller-based virtualization, which to date has been the most popular form of storage virtualization implemented by vendors and technology companies, as well as being deployed in the real world by customers. We'll review some potential use cases, look at how storage virtualization is commonly implemented, and consider some potential pitfalls. We'll finish the chapter by exploring the most recent form of storage virtualization to hit the market: software-defined storage. We'll examine some examples, pros, cons, and potential futures for software-defined storage, which promises to be a disruptive architecture.

What Storage Virtualization Is

Storage virtualization is one of those terms that means many things to many people. Ask five people what it means, and you could easily get five different answers. This isn't because storage virtualization is immature or still undefined as a technology. It's more likely because virtualization happens at just about every layer of the storage stack, and new forms of storage virtualization are coming along all the time.

At its highest level, virtualization is the abstraction of physical devices to logical, or virtual, devices. However, storage virtualization (as well as other forms of virtualization) often goes a step further by virtualizing devices that have already been virtualized. With this in mind, let's quickly look at a common storage stack with applications sitting at the top and physical storage media at the very bottom, and work our way down the stack via the route an application I/O would take. As we do this, I'll point out some of the many areas where virtualization takes place. Figure 9.1 helps make things a little clearer.

FIGURE 9.1 Levels of virtualization in traditional SAN storage


Application speaks to filesystem
/
boot
usr
etc
home
bin
lib
npoulton
lpayne
File systems laid out over logical volumes/partitions
Virtualization
Multiple LUNs configured as a single logical volume
Logical Volume
Virtualization
Logical units (LUN)
Virtualization
Storage network/bus virtualized
Virtualization
Volumes pooled together into storage pools
Virtualization
Physical drives carved into RAID-protected volumes
Virtualization
Simplified LBA view of physical disk drive
Virtualization
Disk drive specifics (platters, tracks, sectors)

Applications usually speak to filesystems or databases, which sit on top of logical volumes. These logical volumes are often made by virtualizing multiple discrete LUNs into a single virtual/logical volume. The underlying LUNs might be accessed over a SAN, which itself is carved up into zones, which are basically virtual SCSI buses. On the storage array, the LUNs themselves are virtual devices defined in cache memory. Beneath these virtual LUNs are *storage pools*, which are virtual pools of storage that are sometimes created from multiple RAID groups. Underneath the RAID groups sit multiple physical drives, but even these physical drives are virtualized into a simple logical block address (LBA) space by their on-board controllers in order to hide the complexity of the inner workings. Every step described here can be considered virtualization.

As shown in Figure 9.1, lower layers in the stack are often virtualized as we move up the stack, resulting in layer upon layer of virtualization. The net result is that although your application, database or filesystem might think it is talking directly to a dedicated locally attached physical disk drive, it is absolutely not! It is talking to a virtual device that has been virtualized many times at many layers in the stack in order to deliver higher performance and higher availability.

NOTE

Because of the multiple layers of virtualization that occur in order to provide an application with persistent storage, no application, filesystem, volume manager, or even operating system has any clue where on disk its data is stored. They may have an address in the local filesystem, but that maps to an address on a logical volume, which maps to an address on a LUN, which maps to pages in a pool, which map to blocks on a logical device on a RAID group, which map to LBA addresses on multiple physical drives, which finally map to sectors and tracks on drive platters. The point is that application, filesystem, and operating system tools have absolutely no idea on which sectors their data lives on the underlying storage media! Don't let this keep you awake at night worrying. These layers of virtualization have been in use for many years and are well and truly field tested and known to be sound. And as you will find out, these layers of virtualization bring massive benefits to the game.

Despite the numerous levels of virtualization previously described, these tend to be so established and deeply embedded that we often forget that they exist and hardly ever consider them as storage virtualization.

The SNIA Shared Storage Model

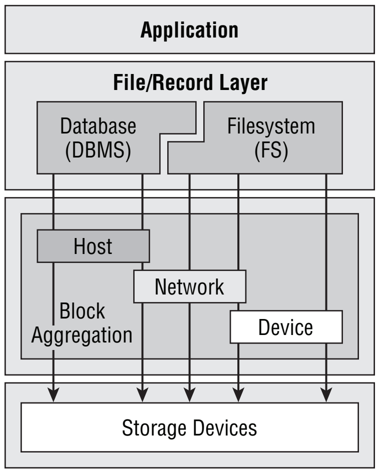
Before we dive into the various types of storage virtualization technology in the market, it's probably a good idea to have a quick look at the SNIA-defined *Shared Storage Model (SSM)*.

NOTE As noted earlier in the book, SNIA is the Storage Networking Industry Association, which is a nonprofit trade association dedicated to advancing storage-related standards, technologies, and education. It is made up of companies that have an interest in advancing storage technologies and standards.

The SNIA SSM is a lot like the better-known Open Systems Interconnection (OSI) seven-layer networking model referred to so often in the network world. Although the SNIA SSM is far less famous and far less referenced than the OSI model, it can be quite useful when describing storage concepts and architectures. In this chapter, we'll use the SSM as a reference point when describing different types of storage virtualization.

Figure 9.2 shows the SNIA Shared Storage Model.

FIGURE 9.2 SNIA Shared Storage Model



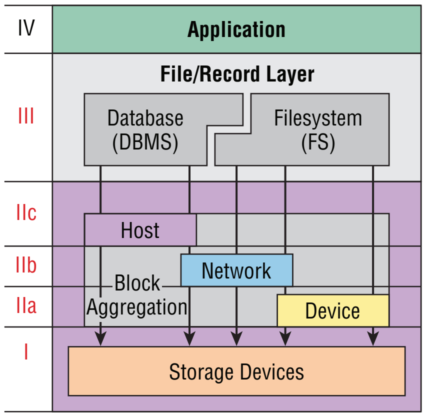
As you can see in Figure 9.2, the SNIA SSM is a pretty simple layered model. At the bottom are the lowest-level components such as disk drives, solid-state media, and tapes. At the top is where higher-level constructs such as filesystems and databases are implemented.

Technically speaking, the SSM is divided into three useful layers under the application layer:

- File/Record layer
- Block Aggregation layer
- Storage Devices layer

In the past, most of the interesting storage virtualization action has occurred at the Block Aggregation layer. But as you can see, there are three discrete sublayers within that layer. Figure 9.3 takes a closer look at the SSM and breaks the layers down a little more.

FIGURE 9.3 Further breakdown of the SNIA Shared Storage Model



The layers are as follows:

- IV. Application
- III. File/Record layer
  - IIIb. Database
  - IIIa. Filesystem
- II. Block Aggregation layer, with three function-placements:
  - IIc. Host
  - IIb. Network
  - IIa. Device
- I. Storage Devices

NOTE The SNIA uses Roman numerals when identifying the layers of the SSM stack. This is done so that they are never confused with the layers of OSI network model.

Based on the SSM layered model, the SNIA defines three *types* of storage virtualization:

- Host-based virtualization
- Network-based virtualization
- Storage-based virtualization

All of these occur at the Block Aggregation layer of the SSM.

Let's have a quick look at each of these.

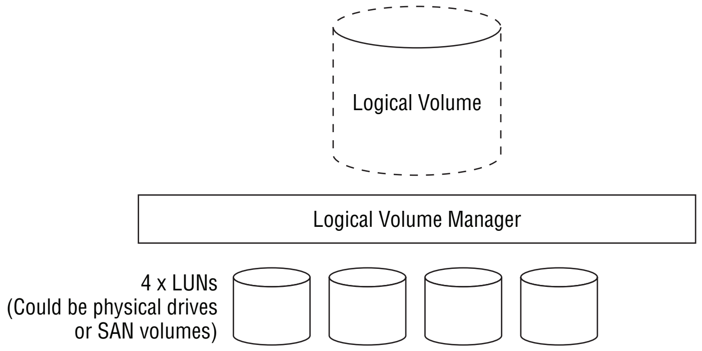
Host-Based Virtualization

Host-based storage virtualization is usually implemented via a *logical volume manager (LVM)*, often referred to as just a *volume manager*. Volume managers work with block storage—DAS or SAN—and virtualize multiple block I/O devices into logical volumes. These logical volumes are then made available to higher layers in the stack, where filesystems are usually written to them.

The logical volumes can be sliced, striped, concatenated, and sometimes even software-RAID protected. From the perspective of storage virtualization, they take devices from the layer below and create new virtual devices known as *logical volumes*. These logical volumes often have superior performance, improved availability, or both.

Figure 9.4 shows a simple volume manager creating a single logical volume from four LUNs presented to it from a SAN-attached storage array.

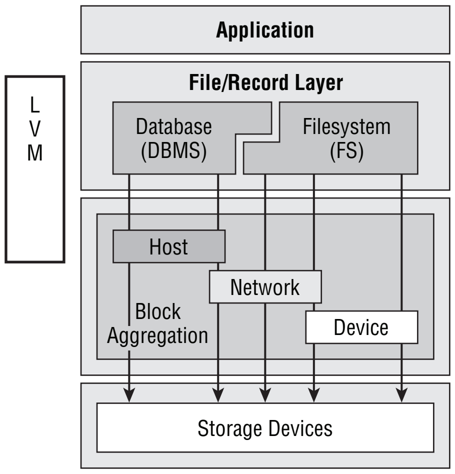
FIGURE 9.4 Logical volume manager creating a logical volume



Although definitely possible, and sometimes useful, volume managers are not often used to create software-RAID-protected logical volumes. This is normally due to the potential performance overhead associated with creating parity-based software RAID. Logical volumes created by a host-based volume manager also tend to be limited in scalability due to being host centric—they can't easily be shared between multiple hosts.

Figure 9.5 shows how host-based volume managers map to the SNIA SSM. Volume managers are host-based software that manipulate block storage devices and aggregate them as new block devices (for example, by taking two block devices and joining them together to form a single mirrored volume). These volumes are then utilized by databases and filesystems.

FIGURE 9.5 LVM mapping to the SNIA Shared Storage Model



In the real world, people generally don't consider LVM functions as storage virtualization. To be fair, this is probably because volume managers have been around and doing this kind of host-based virtualization a lot longer than we have been using the term *storage virtualization*.

Network-Based Virtualization

Network-based (or SAN-based) storage virtualization has been tried and failed. There were a few solutions, the best known probably being EMC Invista, but most, if not all, have gone the way of the dinosaurs.

Conceptually speaking, storage virtualization at the network layer requires intelligent network switches, or SAN-based appliances, that perform functions such as these:

- Aggregation and virtualization of storage arrays
- Combining LUNS from heterogeneous arrays into a single LUN
- Heterogeneous replication at the fabric level (replicating between different array technologies)

All of these were promoted and raved about by vendors and the SNIA. All were attempted, and most have failed. They looked good on paper but, for the most part, never proved popular with customers.

For the record, these devices tended to be out-of-band asynchronous solutions.

Storage-Based Virtualization

Most people refer to *storage-based virtualization* as *controller-based virtualization*. And controller-based virtualization is the predominant form of storage virtualization currently seen in the real world. Because of this, you will look in depth at controller-based virtualization later in the chapter. But in a nutshell, advanced storage arrays are able to attach to other downstream storage arrays, discover their volumes, and use them the same way that they utilize their own internal disks, effectively virtualizing the downstream arrays behind them.

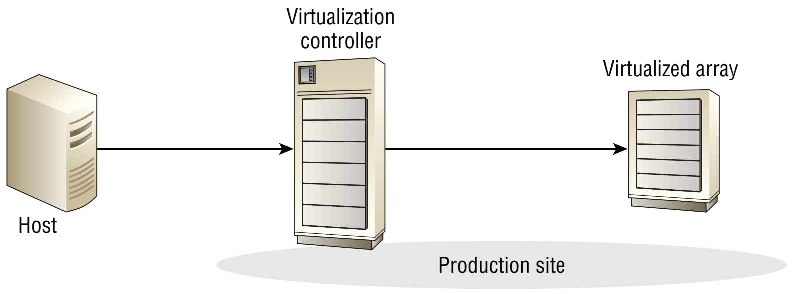
In-Band and Out-of-Band Virtualization

The SNIA categorizes storage virtualization as either in-band or out-of-band.

In *in-band virtualization*, the technology performing the virtualization sits directly in the data path. This means that *all* I/O—user data and control data—passes through the technology performing the virtualization. A common example in the real world is a storage controller that sits in front of and virtualizes other storage controllers. The controller that performs the virtualization sits directly in the data path—in between the host issuing the

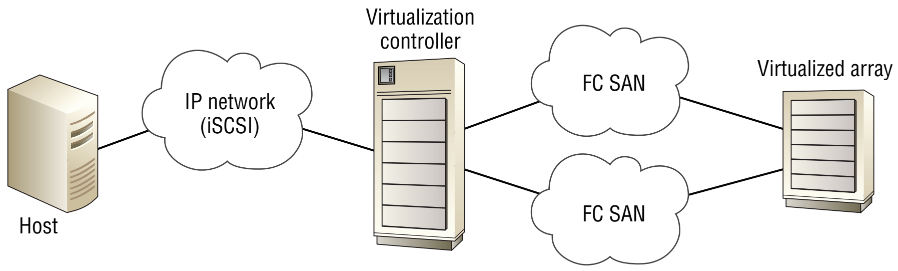
I/O and the storage array that it is virtualizing. As far as the host is concerned, the controller performing the virtualization is its *target* and looks and feels exactly like a storage array (in this case, it is, only it's not the ultimate destination for the data). Whereas from the perspective of the storage array that is being virtualized, it appears as a host. Figure 9.6 gives a high-level view of this type of configuration.

FIGURE 9.6 In-band storage virtualization



There are several advantages to this in-band approach to virtualization. One major advantage is that no special drivers are required on the host, and in the real world this can be a massive bonus. Another advantage is that in-band virtualization can be used to bridge protocols, such as having the device that performs the virtualization connected to FC storage arrays on the backend but presenting iSCSI LUNs out of the front-end. Figure 9.7 shows a configuration where a device performing in-band virtualization is bridging between iSCSI and FC.

FIGURE 9.7 In-band storage virtualization device performing bridging



Out-of-band virtualization, sometimes referred to as *asymmetric,* usually has the metadata pass through a virtualization device or appliance and usually requires special HBA drivers and agent software deployed on the host. It is far less popular and far less deployed than in-band solutions.

Why Storage Virtualization?

So what are the factors that drive the popularity of storage virtualization? Storage virtualization attempts to address the following challenges:

- Management
- Functionality
- Performance
- Availability
- Technology refresh/migrations
- Cost

On the management front, storage virtualization can help by virtualizing multiple assets—be they disk drives or storage arrays—behind a single point of virtualization. This single point of virtualization then becomes the single point of management for most day-to-day tasks. For example, virtualizing three storage arrays behind a Hitachi VSP virtualization controller allows all four arrays to be managed by using the Hitachi VSP management tools. And as with all good virtualization technologies, multiple technologies from multiple vendors can be virtualized and managed through a single management interface.

On the functionality front, it is possible to add functionality to the storage devices being virtualized. A common example is when virtualizing low-tier storage arrays behind a higher-tier array. The features and functionality, such as snapshots and replication, of the higher-tier array (the array performing the virtualization) are usually extended to the capacity provided by the lower-tier array.

On the performance front, storage virtualization can improve performance in several ways. One way is by virtualizing more disk drives behind a single virtual LUN. For example, if your LUN has 16 drives behind it and needs more performance, you can simply increase the number of drives behind the virtual LUN. Another way that storage virtualization commonly assists with server virtualization is by virtualizing smaller, lower-performance arrays behind a high-performance, enterprise-class array with a large cache, effectively extending the performance of the large cache to the arrays being virtualized.

On the availability front, quite low in the stack, RAID groups are used to create RAID-protected LUNs that look and feel like a physical block access disk drive but are actually made up of multiple physical disk drives formed into a resilient RAID set. Array-based replication and snapshot technologies also add to the protection provided by virtualization.

On the technology refresh front, controller-based virtualization is often used to smooth out storage migrations. This helps volumes presented to a particular host to be nondisruptively migrated from one storage array to another without downtime to the host or interruption to I/O.

From a cost perspective, storage virtualization can assist by virtualizing cheaper, lower-end storage arrays behind higher-performance arrays. This brings the performance benefits of the higher-performance array to the capacity of the virtualized array. It also extends the features—such as replication, snapshots, deduplication, and so on—of the high-end array to the capacity of the lower-cost array.

WARNING

Take all of these benefits of virtualization with a pinch of salt. They are all plausible and implemented in the real world, but they are also all subject to caveats. As a quick example, virtualizing low-end storage arrays behind a high-performance array can extend the performance of the high-performance array to the capacity provided by the low-end array, but only up to a point. If you try to push it too far, performance of the high-end array can start to suffer.

NOTE

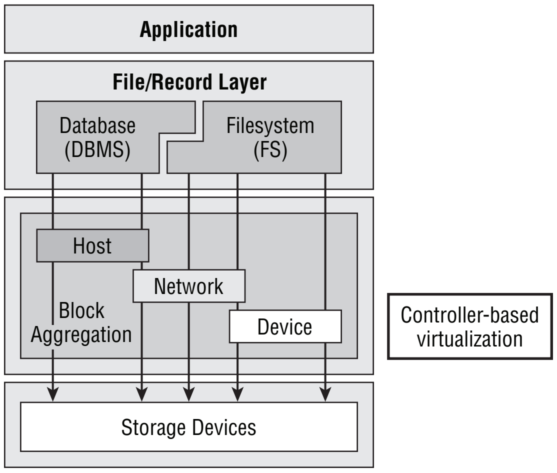
As always, all cost savings depend on how much you pay. Virtualization devices and licenses are seldom cheap or free.

Enough of the theory. Let's take a closer look at the most popular form of storage virtualization: controller-based virtualization.

Controller-Based Virtualization

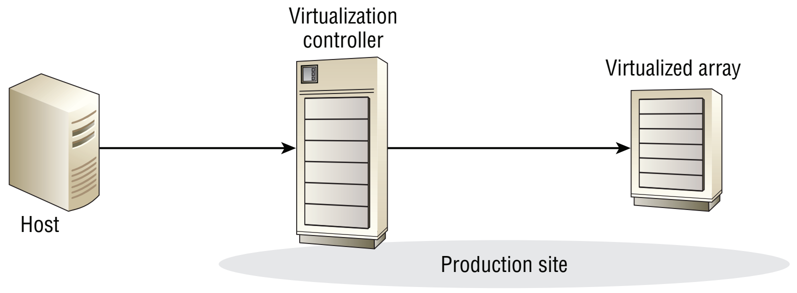
First up, let's map this to the SSM model. *Controller-based virtualization* occurs at the device layer within the block aggregation layer, and is a form of *block-based in-band virtualization*. This is shown in Figure 9.8.

FIGURE 9.8 Controller-based virtualization mapped to the SSM



At a high level, controller-based virtualization is the virtualization of one storage array behind another storage array, as shown in Figure 9.9.

FIGURE 9.9 Controller-based virtualization



NOTE In a controller virtualization setup, one array acts as the master and the other as the slave. We will refer to the master as the *virtualization controller* and the slave as the *virtualized array* or *array being virtualized.*

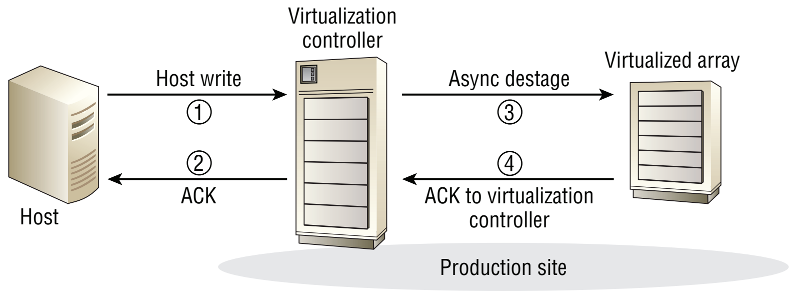
In a controller-based virtualization configuration, the virtualizing controller sits *in front* of the array being virtualized. As far as hosts issuing I/O are concerned, the virtualization controller is the *target,* the ultimate end point to which data is sent. The host needs no special driver or agent software and is blissfully unaware that the array it is talking to is effectively a middleman and that its data is in fact being sent on to another storage array for persistent storage.

The virtualization controller is where all of the intelligence and functionality resides. The array being virtualized just provides dumb RAID-protected capacity with a bit of cache in front of it. None of the other features and functions that might be natively available in the virtualized array are used, and they therefore don't need to be licensed.

In most controller-based virtualization setups, when the host issues a write I/O, it issues I/O to the virtualization controller, which caches the I/O and returns the ACK to the host. The write I/O is then lazily transferred to the array being virtualized, according to the cache destage algorithms of the virtualization controller. The virtualization controller usually imitates a host when talking to the arrays virtualized behind it and issues the same write data to the virtualized array, which in turn caches it and commits it to disk according to its own cache destage algorithm. This is shown in Figure 9.10.

Because the virtualizing array usually imitates a host when talking to the array virtualized behind it, no special software or options are required on the arrays being virtualized. As far as they are concerned, they are just connected to another host that issues read and write I/O to it. For example, the HDS VSP platform supports controller-based virtualization and imitates a Windows host when talking to the arrays virtualized behind it.

FIGURE 9.10 Write I/O to a virtualized storage array



Vendors such as HDS and NetApp adopted the term *storage hypervisor* to describe their controller-based virtualization platforms. The description works, up to a certain point.

Next let's take a look at a typical controller-based storage virtualization configuration and walk through the high-level steps in configuring it.

Typical Controller Virtualization Configuration

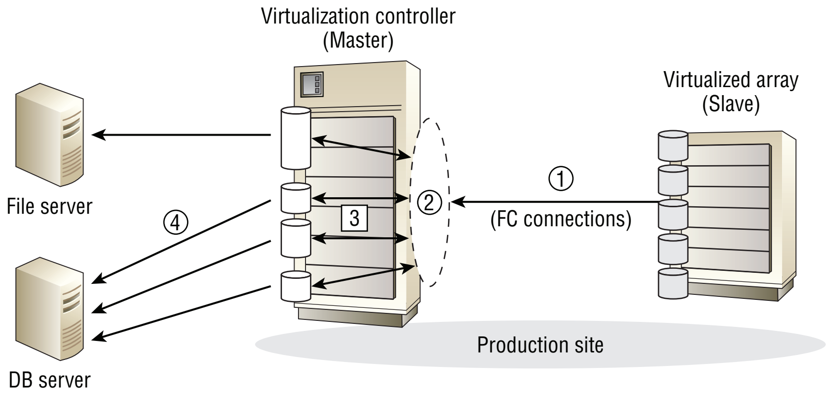
In this section, you'll walk through the typical high-level steps required to configure controller-based virtualization. These steps are based around configuring controller-based virtualization on a Hitachi VSP array. Even though the steps are high level, they can solidify your understanding of how to configure a typical controller-based virtualization setup.

Configuring the Array Being Virtualized

Typically, the first thing to do is configure the array that you will be virtualizing. On this array, you create whatever RAID-protected volumes that you would usually configure, and you present them as SCSI LUNs on the front-end ports. You need to configure LUN masking so that the WWPNs of the virtualization controller have access to the LUNs. These LUNs don't need any special configuration parameters and can usually use normal cache settings. It is important that you configure any host mode options for these LUNs and front-end ports according to the requirements of the virtualization controller. For example, if the virtualization controller imitates a Windows host, you need to make sure that the LUNs and ports are configured to expect a Windows host.

For example, Figure 9.11 shows a virtualized array with 100 TB of storage configured as five 2 TB LUNs presented to four front-end ports. The front-end ports are configured to Windows host mode, and LUN masking is set to allow the WWPNs of the virtualization controller ports.

FIGURE 9.11 Configuring an array to be virtualized



1. LUNs on the virtualized array are presented out to the WWPNs of the virtualization controller.
2. The virtualization controller claims these LUNs and uses them as storage. In this example, they are formed into a pool.
3. Volumes are created from the pool created in step 2.
4. The volumes created in step 3 are presented out of the front-end ports to hosts as LUNs.

WARNING

It is vital that no other hosts access the LUNs that are presented from the virtualized array to the virtualization controller. If other hosts access these LUNs while the virtualization controller is accessing them, the data on those LUNs will be corrupted. The most common storage virtualization configurations have all the storage in the virtualized array presented to the virtualization controller. This way, there is no need for any hosts to connect directly to the virtualized array. It is, however, possible to configure some LUNs on the virtualized arrays for normal host access, with other LUNs used exclusively by the virtualization controller. The important point is that no single LUN should be accessed by both.

Configuring the Virtualization Controller

After configuring the array to be virtualized, next you configure the virtualization controller. Here you will configure two or more front-end ports—yes, front-end ports—into virtualization mode. This virtualization mode converts a port from its normal target mode to a special form of initiator mode that allows the ports to connect to the array being virtualized and discover and use its LUNs. In order to keep things simple, these ports usually emulate a standard Windows or Linux host so that the array being virtualized doesn't need to have any special configuration changes made. This keeps interoperability and support very simple.

Connecting the Virtualization Controller and Virtualized Array

Connectivity is usually FC and can be either direct attach or SAN attach. Due to the critical nature of these connections, many people opt for direct attach in order to keep the path between the two arrays as simple and clean as possible.

When the virtualization controller is connected to the array being virtualized, it performs a standard PLOGI. Because its WWPN has been added to the LUN masking list on the array being virtualized, it will discover and claim the LUNs presented to it the same way it would use capacity from locally installed drives. The common exception is that the virtualization controller usually does not apply RAID to the discovered LUNs. Low-level functions such as RAID are still performed by the virtualized array.

From this point on, the LUNs discovered can be used by the virtualization controller in exactly the same way that it uses its own internal volumes. By that, we mean that the external volumes can be used to create pools and LUNs.

Figure 9.11 shows a virtualized array presenting five LUNs to the virtualization controller. The virtualization controller logs in to the virtualized array, discovers and claims the LUNs, and forms them into a pool. This pool is then used to provide capacity for four newly created volumes that are presented out of the front-end of the virtualization controller to two hosts as SCSI LUNs. The two hosts have no idea that the storage for the LUNs is virtualized behind the array it is talking to. In fact, the host thinks it is talking to physical disk drives that are physically installed in the server hardware that it is running on!

The local storage in the virtualization controller is referred to as *internal storage*, whereas the storage in the array being virtualized is often referred to as *external storage*.

Putting It All Together

Exercise 9.1 recaps the previous sections in a more succinct format.

EXERCISE 9.1

Configuring Storage Virtualization

These steps outline a fairly standard process for configuring storage virtualization. The exact procedure in your environment may vary, so be sure to consult your array documentation. If in doubt, consult with your array vendor or channel partner. We walk through a high-level example here to help you understand what is going on.

1. On the array being virtualized, perform the following tasks:
    1. Carve the backend storage into RAID-protected volumes. Normally, large volumes, such as 2 TB, are created and presented out of the front as FC LUNs.
    2. Present these LUNs on the front-end ports and mask them to the WWPNs of the virtualization controller.
    3. Configure any standard cache settings for these LUNs.
    4. Configure any host mode settings required by the virtualization controller. For example, if your virtualization controller emulates a Windows host, make sure you configure the host mode settings appropriately.

2. On the virtualization controller, configure a number of front-end ports as external ports (virtualization mode). You should configure a minimum of two for redundancy.
3. Now that the basic tasks have been performed on the virtualization controller as well as the array being virtualized, the arrays need to be connected. These connections can be either direct attach or SAN attach. The important thing is that they can see each other and communicate.
4. Once the arrays are connected and can see each other, use the virtualization controller's GUI or CLI to discover the LUNs presented to it from the virtualized array.

These newly discovered volumes can then be configured and used just like any other disk resources in the virtualization controller, such as using them to create pools or exported LUNs.

Now that we've described how controller-based virtualization works, let's look more closely at some of the potential benefits, including the following:

- Prolonging the life of existing assets
- Adding advanced functionality to existing assets
- Augmenting functionalities such as auto-tiering
- Simplifying migration

Prolonging the Life of an Existing Array

One of the touted benefits of virtualizing an existing storage array is to prolong its life. The idea is that instead of turning off an old array (because it's out-of-date and doesn't support the latest advanced features you require), you decide to give it a brain transplant by virtualizing it behind one of your newer arrays, giving it a new lease of life.

This idea works fine on paper, but the reality tends to be somewhat more complicated. Even in a difficult economy with diminishing IT budgets, arrays are rarely virtualized to prolong their life.

Some of the complications include the following:

- Having to keep the virtualized array on contractual maintenance congruent to the environment it will continue to serve. It is not the best idea in the world to have the old virtualized array on a Next Business Day (NBD) support contract if it's still serving your tier 1 mission-critical apps, even if it is virtualized behind another newer array.
- Multi-vendor support can be messy and plagued with finger-pointing, where each vendor blames the other and you get nowhere fast! This is made all the more challenging if the array you are virtualizing is old, as the vendor is usually not keen for you to keep old products on long-term maintenance and they generally stop releasing firmware updates and so on.

Experience has shown that although virtualizing an array to prolong its life works on paper, reality tends to be somewhat more messy.

Adding Functionality to the Virtualized Array

It's a lot more common for customers to buy a new tier 1 enterprise-class array along with a new tier 2/3 midrange array and to virtualize the midrange array behind the enterprise-class array. This kind of configuration allows the advanced features and intelligence of the tier 1 array to be extended to the capacity provided by the virtualized tier 2/3 array. Because the array performing the virtualization treats the capacity of the virtualized array the same way it treats an internal disk, volumes on the virtualized array can be thin provisioned, deduplicated, replicated, snapshotted, tiered, hypervisor offloads-supported, you name it.

This kind of configuration is quite popular in the real world.

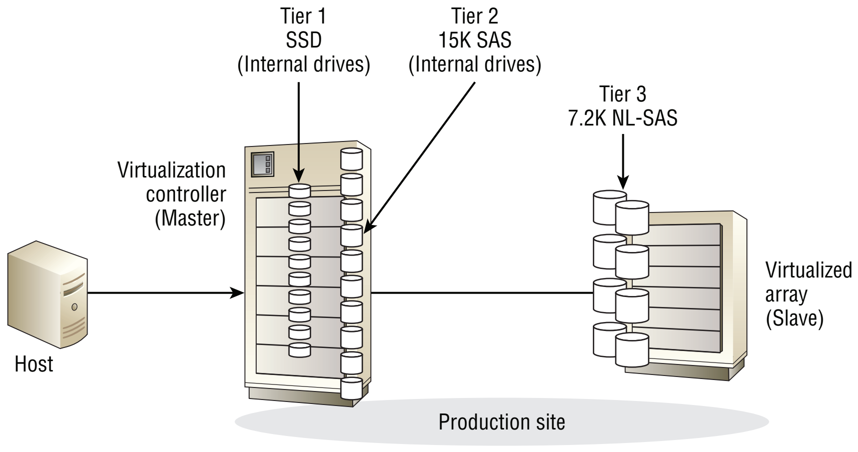
Storage Virtualization and Auto-tiering

All good storage arrays that support controller-based virtualization will allow the capacity in a virtualized array to be a tier of storage that can be used by its auto-tiering algorithms. If your array supports this, then using the capacity of your virtualized array as a low tier of storage can make sound financial and technical sense.

Because the cost of putting a physical disk in a tier 1 array can make your eyes water—even if it's a low-performance disk such as 4 TB NL-SAS—there is solid financial merit in populating only the internal drive slots of a tier 1 array with high-performance drives. Aside from the high cost of populating internal drive slots, these internal drive slots also provide lower latency than capacity that is accessed in a virtualized array. So it stacks up both financially and technically to keep the internal slots for high-performance drives and use virtualized capacity as a cheap and deep tier.

This leads nicely to a multitier configuration that feels natural, which is shown in Figure 9.12.

FIGURE 9.12 Storage virtualization and auto-tiering



There is no issue having the hot extents of a volume on *internal* flash storage, the warm extents on *internal* high-performance SAS, and the coldest and least frequently accessed extents down on 7.2 K or 5.4 K NL-SAS on an externally virtualized array. This kind of configuration is widely deployed in the real world.

Of course, this all assumes that your virtualization license doesn't cost so much that it destroys any business case for storage virtualization.

Additional Virtualization Considerations

There is no getting away from it: controller-based virtualization adds a layer of complexity to your configuration. This is not something to run away from, but you should definitely be aware of it before diving in head first.

Real World Scenario

Complications That Can Arise Due to Storage Virtualization

On the topic of complexity, one company was happily using storage virtualization until a senior storage administrator made a simple mistake that caused a major incident. The storage administrator was deleting old unused *external* volumes—volumes that were being virtualized. However, the virtualization controller gave volumes hexadecimal IDs, whereas the virtualized array gave volumes decimal IDs. To cut a long story short, the administrator got his hex and decimal numbers mixed up and deleted a whole load of the wrong volumes. This caused a lot of systems to lose their volumes, resulting in a long night of hard work for a lot of people. This minor complexity of hex and decimal numbering would not have existed in the environment if storage virtualization was not in use.

Depending on how your array implements storage virtualization and depending on how you configure it, you may be taking your estate down a one-way street that is hard to back out of. Beware of locking yourself into a design that is complicated to change if you later decide that storage virtualization is not for you.

If not designed and sized properly, performance can be a problem. For example, if you don't have enough cache in your virtualization controller to deal with the workload and additional capacity provided by any virtualized arrays behind it, you may be setting yourself up for performance problems followed by costly upgrades. Also, if the array you are virtualizing doesn't have enough drives to be able to cope with the data being thrown at it by the virtualization controller, it can quickly cause performance problems.

Real World Scenario

The Importance of Planning Your Storage Virtualization Properly

I previously worked with a company that hadn't sized or configured their live production storage virtualization solution appropriately. They made two fundamental mistakes.

First, they didn't provide enough performance resources (on this occasion, not enough disk drives) in the array being virtualized. This meant that at peak times, the virtualized array was woefully incapable of receiving the large amounts of data the virtualization controller was sending it. This resulted in data backing up in the virtualization controller's cache and quickly brought the virtualization controller and all hosts connected to it to a grinding halt.

Their problem wasn't helped by the fact that they had configured the cache on the virtualization controller to be used as a *write-back cache* for the virtualized volumes. This meant that as the host writes destined for the virtualized array came into the virtualization controller, they ACKed very quickly and cache started filling up. However, as cache got fuller, it became important to get the data out of cache and onto disk, but the disk in this instance was on the virtualized array. And as we mentioned earlier, the virtualized array was undersized, and the data couldn't be pumped out of the cache fast enough, resulting in a huge cache-write-pending situation that brought the virtualization controller to its knees.

Finally, the cost of a virtualization setup can be an issue if you don't know what you're getting yourself into. Virtualization licenses rarely come for free, and their cost must be factored into any business case you make for controller-based virtualization!

Software-Defined Storage

Software-defined storage (SDS) is another form of storage virtualization. However, to make things more complicated here, there is no formal definition for software-defined storage, and it has been said by many that "there is no term as badly defined as *software defined*."

For a lot of vendors, it's little more than a marketing buzzword, and every vendor is desperately trying to prove that their products are software defined. However, despite the lack of an agreed-upon definition, *there are massive design and architectural changes afoot that* *promise to redefine not only storage, but the entire data center!* And there absolutely are vendors out there shipping real software-defined storage products!

We will talk about some next-generation storage virtualization technologies that, in my opinion, fit as software-defined storage. Either way, they are forms of storage virtualization that differ from what we have already talked about. They're very cool!

SDS Explained

In order to better explain software-defined storage, it's useful to first take a look at traditional non-software-defined storage. Then we have something to compare to.

Traditional Non-Software-Defined Storage

In traditional non-SDS storage arrays, the intelligence is tightly coupled with physical hardware. It is tightly coupled with the hardware *controllers*, which in turn are usually tightly coupled with the physical drives and the rest of the *backend*. Put another way, the firmware, which provides the intelligence and all the data services, will run only on specific hardware controllers from the same vendor. You can't take the microcode from one vendor's array and run it on the controller hardware of another vendor's array. You can't even take it and run it on industry-standard server hardware platforms such as HP ProLiant and Cisco UCS.

In traditional non-software-defined storage, the software and the hardware are married in a strictly monogamous relationship!

NOTE When talking about *intelligence,* we're referring to everything—high availability, snapshots, replication, RAID, deduplication, thin provisioning, hypervisor offloads, you name it. In traditional non-software-defined storage architectures, it's all implemented in microcode that will run only on specific hardware. In software-defined storage, this intelligence can run on any industry-standard hardware or hypervisor.

A lot of traditional arrays provide high availability (HA) via dual-controller architectures. Take away the dual controllers—specific hardware with specific controller interconnects—and you can no longer implement HA.

As if the integration between intelligence and hardware wasn't already tight enough for traditional arrays, some take it a step further and implement some of their intelligence *in* the hardware, as is the case with custom silicon such as ASICs and FPGAs. This is hardware and software integration at its tightest.

Basically, with traditional storage arrays, the software is useless without the hardware, and the hardware is useless without the software. The two are welded together at the factory!

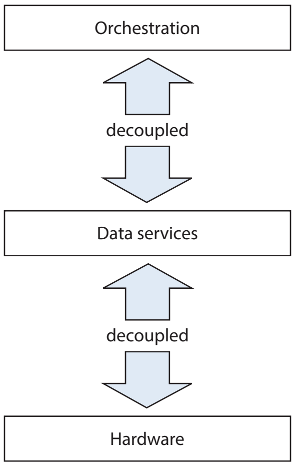
This tight integration makes it hard to take these architectures and rework them for more-flexible SDS solutions. But, hey, who says they need to? These tightly coupled architectures are still running businesses across the world.

The SDS Way

SDS is the polar opposite of traditional arrays and their tightly coupled intelligence and hardware. SDS is all about decoupling the hardware and the intelligence. It takes all of the value out of the hardware and shovels it into software. That makes the hardware an almost irrelevant, cheap commodity.

From an architectural perspective, software-defined storage sees storage in three layers—orchestration, data services, and hardware—as shown in Figure 9.13. None of these layers are tightly integrated with each other, meaning that you can mix and match all three, giving you more flexibility in choosing your hardware, your data services, and your orchestration technologies.

FIGURE 9.13 Three layers of software-defined storage



NOTE You may hear people refer to software defined storage (SDS) as separating the *control plane* from the *data plane*. This terminology is more popular and better suited to the network world, where the concepts of control planes and data planes are more widely used. Also, in the networking world, there is more control plane commonality as devices from different vendors interact with each other far more often than in the storage world. However, the principle is the same; it is referring to this decoupling of intelligence from hardware.

There are several key aspects to SDS. One of the most prominent features is the use of virtual storage appliances (VSAs) to reduce the strain on hardware. APIs are a convenient feature of working with SDS. SDS is easily scalable and is compatible with the cloud and other innovations. Let's look at each of these points a little more closely.

One popular example of ripping the intelligence out of the hardware—which is certainly a form of software-defined storage—is the *virtual storage appliance (VSA)*. A VSA is an instance of a fully fledged intelligent storage controller implemented as pure software in the form of a virtual machine (VM) that runs on industry-standard hypervisors, which in turn run on industry-standard x64 hardware.

VSAs can utilize local disks in the physical servers they run on. They can also utilize capacity from DAS disk trays or even shared storage such as iSCSI and FC. Many of the VSA vendors recommend large, dumb JBOD (just a bunch of disks) storage attached to them for capacity.

Running storage controllers—the intelligence—as virtual machines opens up a whole new world of possibilities. For example, if you need to add resources to a VSA, just have the hypervisor assign more CPU (vCPU) or RAM (vRAM) to it. When you no longer need the additional resources assigned to the VSA, simply reduce the amount of vCPU and vRAM assigned. We could even go as far as to say that the entire stack (compute, storage, networking, applications, and so on) could be intelligent enough to ramp up resources to the VSA at times of high demand on the VSA and then to take them away again when the VSA no longer needs them. A simple example might be an overnight post-process deduplication task that is CPU intensive. When the job kicks in, either run an associated script to add more vRAM and vCPU or allow the system to dynamically assign the resources. Then when the dedupe task finishes, take the resources back. Now that's what I call dynamic!

These VSAs can run on the same hypervisor instance as your hosts and applications, bringing the storage closer to the application.

Established vendors, such as HP, as well as startup vendors including the likes of Nutanix, Nexenta, ScaleIO, SimpliVity, and others, have technologies that utilize the VSA software-defined storage model.

SDS is also API driven. In fact, more than that, it is *open* API driven, including RESTful APIs. RESTful APIs use the following simple verbs to perform functions: GET, POST, PUT, and DELETE. A major aim with SDS is for users to interact with the system via APIs, as well as the system itself using APIs to make its own internal calls. It is all about the APIs with software-defined storage. Contrast this with more-traditional non-SDS storage, where administration was handled via a CLI and usually very poor GUIs.

SDS is scale-out by nature. Take the simple VSA example again. If you need more storage controllers, just spin up more VSAs. These can be managed by the common industry standard APIs mentioned earlier, allowing for simpler, more centralized management. It is no longer necessary to log on to each array in order to configure it.

SDS embraces the cloud. As the storage layer is already abstracted and physically removed from the intelligence, it is simpler to implement cloud as a tier or target for storage. Traditional arrays tend to be designed with the storage medium close to the controllers, and the software intelligence on the controllers understands minute detail about the storage medium. SDS takes a totally different approach, where the storage hardware is abstracted, making cloud integration massively simpler and more fundamental. SDS solutions embrace cloud APIs such as REST and offer integrations such as Amazon S3 and OpenStack Swift.

Another benefit of the software-defined model is that the rate of software innovation is far higher than that of hardware innovation. Basically, it is easier, faster, and cheaper to innovate in software than it is to innovate in hardware. So why slow down your innovation cycles by pinning innovation to hardware upgrades and releases?

Will Software-Defined Storage Take Over the World?

While the concept and some of the implementations of SDS are really interesting and absolutely have their place, they most likely will exist alongside traditional storage arrays for a very long time.

After all, it was not that long ago that traditional, tightly coupled architectures were all the rage, and terms like *appliance, turnkey, and prepackaged* were in vogue. Now they are blasphemous if you listen to those who have planted their flag in the SDS world. But let's not forget that the traditional approach of fully testing, burning in, and slapping a "designed for XYZ" logo on the side of a traditional array has its advantages. For one thing, you know that the vendor has tested the solution to within an inch of its life, giving you the confidence to run your tier 1 business-critical applications on it. You also have a single vendor to deal with when things do go wrong, instead of having several vendors all blaming each other.

NOTE

Interestingly enough, some vendors ship SDS solutions as software-only. Others sell SDS solutions as software and hardware packages that include both the controller intelligence and the hardware. The main reason for this is supportability. Although the software (intelligence) and hardware are sold as a single package here, they do not *have* to be. This is purely to make support simpler because the vendor will have extensively tested the software running on their hardware and will have it running in their labs to assist with support calls.

At the end of the day, established companies around the world aren't going to give up on something that has served them well for many years. They will likely start testing the water with SDS solutions and implement them in lab and development environments before trusting their mission-critical systems to them. Newer companies, which are more likely to be highly virtualized with zero investment in existing storage technologies, are far more likely to take a more adventurous approach and place their trust in SDS-based solutions.

As with all things, do your homework and try before you buy!

Chapter Essentials

Host-Based Storage Virtualization Host-based virtualization is usually in the form of a logical volume manager on a host that aggregates and abstracts volumes into virtual volumes called logical volumes. Volume managers also provide advanced storage features such as snapshots and replication, but they are limited in scalability, as they are tied to a single host. Most people don't consider host-based storage virtualization as a true form of storage virtualization.

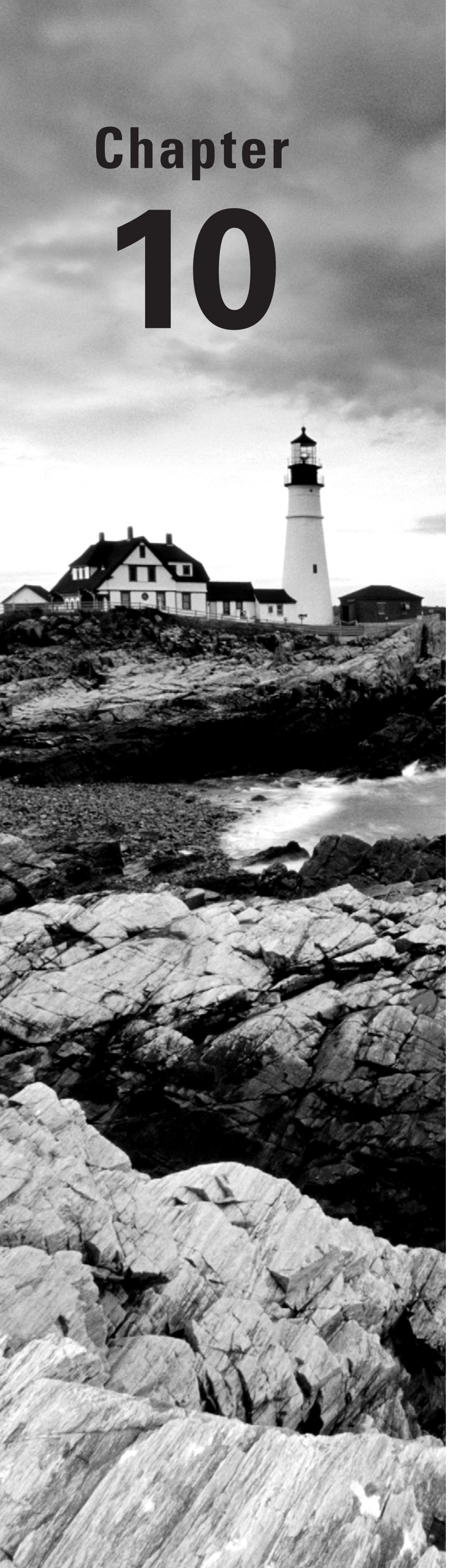
Network-Based (SAN-Based) Storage Virtualization Network-based virtualization is more of a concept than a reality these days, as most of the products in this space didn't take off and have been retired.

Controller-Based Storage Virtualization This is by far the most common form of storage virtualization and consists of an intelligent storage array (or just storage controllers) that can virtualize the resources of a downstream array. Controller-based storage virtualization is a form of in-band, block-based virtualization.

Software-Defined Storage SDS is any storage controller intelligence implemented solely in software that can be installed on industry-standard hypervisors running on industry-standard server hardware that can virtualize and add intelligence to underlying commodity storage.

Summary

In this chapter we covered all of the popular implementations of storage virtualization seen in the real world, and concentrated on the most popular form—controller based virtualization. We covered some of the major use cases for storage virtualization, as well as some of the potential problems you'll encounter if you don't get it right. We then finished the chapter with a discussion of the hot new topic of software defined storage (SDS) and compared and contrasted it with the more traditional approach of tightly integrating storage intelligence with storage hardware.



Capacity Optimization Technologies

TOPICS COVERED IN THIS CHAPTER:

- ✓ **Thin provisioning**
- ✓ **Overprovisioning**
- ✓ **Data compression**
- ✓ **Data deduplication**
- ✓ **Auto-tiering**
- ✓ **Sub-LUN auto-tiering**

Over the past few years, the storage industry has seen a major uptick in the turnout of high-quality capacity optimization technologies. These include thin provisioning, compression, deduplication, and auto-tiering. This trend is due to several factors, including shrinking IT budgets, the need to do more with less, the natural maturation of storage technologies, and a major push from flash storage vendors that need strong capacity optimization technologies in order to make their offerings price-competitive with solutions based on spinning disk. The net result is that there are a whole host of high-quality capacity optimization technologies to choose from.

This chapter presents all of these major capacity optimization technologies. It covers topics such as inline and post-process data deduplication and data compression, as well as thin provisioning, overprovisioning, and tiering. Following our usual approach, you'll dig deep into how these technologies work and are implemented in the real world, as well as mapping some of the technologies to use cases, such as backup and archiving, and primary storage. You'll learn where some of these technologies might not be the best fit, so you don't make mistakes that you could have avoided. The chapter discusses the impacts that capacity optimization technologies have on key principles such as performance, availability, and cost. After reading this chapter, you should be equipped to deploy the right data-efficiency technologies in the right places.

Capacity Optimization Technologies Overview

SNIA uses the term *capacity optimization methods* to refer to all methods used to reduce the space required to store persistent data. This pretty much puts it in a nutshell. Capacity optimization technologies are all about doing more with less: storing more data on less hardware, sweating your storage assets, managing data growth, squeezing every inch of capacity out of media, and the list could go on.

While there may be some other fringe benefits, such as performance improvements, these are very much side-benefits and bonuses. The raison d'etre of capacity optimization technologies is increasing capacity utilization and enabling you to store more data in less space.

When discussing increasing capacity utilization, we'll naturally discuss the technologies that will have a positive impact on financial matters such as return on investment (ROI) and total cost of ownership (TCO). This includes things such as $/TB acquisition cost; reduced data center footprint; reduced network bandwidth (network costs); reduced power and cooling; and a leaner, greener data center.

Performance Implications

Let's take a quick moment to look at some potential performance implications that result from implementing capacity optimization technologies. As with most technologies, capacity optimization technologies can increase or decrease overall storage and system performance. For example, tiering (which we will consider to be a capacity optimization technology) might move data down to lower tiers, decreasing performance, or up to higher tiers, increasing performance. Deduplication similarly can increase or decrease performance. Inline deduplication of data residing on primary storage arrays that are based on spinning disk will *usually* lower both read and write performance, whereas post-process deduplication of virtual machines with a sizable read cache can significantly improve performance. Thin provisioning (TP) can also improve performance by scattering data over very large numbers of drives on the backend of a storage array, but it can also lead to fragmented layout on the backend and negatively impact sequential performance.

We'll cover plenty on the topic of performance as we discuss each capacity optimization technology, but for now the takeaway point should be that capacity optimization technologies can send performance in either direction—up or down. It all depends on how the technology is implemented. So beware.

Types of Capacity Optimization Technologies

Lots of capacity optimization technologies are available—in fact, so many that it can be confusing. To clarify, let's begin with a quick overview of the types of technology, which you'll explore more throughout the chapter:

Thin provisioning is the process of allocating storage capacity on demand rather than reserving mega amounts of space up front just in case you might need it two years later.

Compression, another of the more well-known capacity optimization technologies, is re-encoding of data to reduce its size. This is subtly, but fundamentally, different from deduplication.

Deduplication, another of the more popular and well-known capacity optimization technologies, is the replacement of multiple copies of identical data with references to a single shared copy, all with the end goal of saving space.

Tiering technologies seek to place data on appropriate storage mediums, usually based on frequency of access to the data, such as moving infrequently accessed data down to cheaper, slower storage and putting more frequently accessed data on more-expensive, higher-performance storage. While this doesn't strictly conform to the description of increasing capacity utilization, it does optimize how your capacity is utilized.

There are also various places at which capacity optimization technologies can be implemented: source-based, appliance-based, gateways, target-based, and so on. They can also be implemented inline, post-process, or even somewhere in between. We'll cover most of these options throughout the chapter.

Use Cases for Capacity Optimization Technologies

The use cases for capacity optimization technologies are almost limitless. Technologies that were once considered off-limits because of performance and other overhead are now becoming viable thanks to advances in CPU, solid-state storage, and other technologies.

Over the next few years, we will see capacity optimization technologies implemented at just about every level of the stack and included in most mainstream storage technologies. Backup, archiving, primary storage, replication, snapshots, cloud, you name it, if it's not already there, it will be soon.

There is a lot to cover, so let's press on.

Thin Provisioning

In the storage world, *thin provisioning (TP)* is a relatively new technology but one that has been enthusiastically accepted and widely adopted. A major reason behind the rapid adoption of TP is that it helps save money, or at least avoid spending it. It does this by working on an *allocate-on-demand* model, in which storage is allocated to a volume only when data is actually written, rather than allocating large quantities of storage up front. You'll look more closely at how it works shortly.

This chapter uses the terms *thick volumes* and *thin volumes*, but these concepts are also known as *traditional volumes* and *virtual volumes*.

Thick Volumes

As thick volumes are the oldest and most common type of volumes, let's take a moment to review them to set the scene for our discussion on the more recent and complicated topic of thin volumes.

Thick volumes are simple. You choose a size, and the volume immediately reserves/consumes the entire amount defined by the size. Let's say you create a 1 TB thick volume from an array that has 10 TB of free space. That thick volume will immediately consume 1 TB and reduce the free capacity on the array to 9 TB, even without you writing any data to it. It is simple to understand and simple to implement, but it is a real waste of space if you never bother writing any data to that 1 TB thick volume. In a world with an insatiable appetite for a finite pool of storage, such potential wasting of space is a problem—and it's the main problem with thick volumes that can't go unaddressed.

The problem of thick volumes allowing users to waste storage space is only compounded by the common practice of people wildly overestimating the amount of storage they need, just in case they *may* need it at some point in the future. It's common for storage arrays to be fully provisioned—with all their capacity carved up into thick volumes and presented out to hosts—but for a lot less than half of that capacity to be actually used by the hosts. The other half sits there doing nothing. Imagine a 200 TB storage array that has all 200 TB allocated to hosts as thick volumes, but the total combined data from all hosts written to those thick volumes being in the region of 80 TB. That's wasting more than 100 TB of storage. That kind of ratio of provisioned-to-used capacity is still common in too many places. However, thanks in no small part to thin provisioning, this is becoming less of an issue.

Thin Volumes

Thin volumes work a lot differently than thick volumes. Thin volumes consume little or sometimes no capacity when initially created. They start consuming space only as data is written to them.

NOTE

In most systems, thin volumes do consume a small amount of space upon creation. This space is used so that as a host writes, an area of space is ready for the data to land on without the system having to go through the process of locating free space and allocating it to the thin volume. Not all systems work like this, but many do.

Let's take a quick look at how a 1 TB thin volume works. When we create this 1 TB thin volume, it initially consumes little or no space. If the host that we export it to never writes to it, it will never consume any more space. Let's say an eager developer thinks her new business application needs a new volume to cope with the large amounts of data it will generate and requests a 2 TB volume. As a storage administrator, you allocate a new 2 TB volume from your array that has 10 TB of free space available. However, the application generates only a maximum of 200 GB of data, nowhere near the amount of data expected. If this 2 TB volume had been a thick volume, the instant you created it, it would have reduced your array's available capacity from 10 TB to 8 TB. Most of that would have been wasted, as the application never even comes close to touching the sides of the 2 TB volume. However, as it was a thin volume, only 200 GB was ever written to it. The unused 1.8 TB of space is still available to the array to allocate to other volumes. This is obviously a powerful concept.

Filesystems Don't Always Delete Data

It's vital to understand that because of the way many applications and filesystems work, the amount of space that a filesystem reports as being used does not always match the amount of space that a storage array thinks is used. For example, if you have a 1 TB thin volume, and an application or user writes 100 GB to a filesystem on that volume, this will show up as about 100 GB of data consumed in your 1 TB thin volume. Now assume that your user or application then deletes 50 GB of the 100 GB of data. The filesystem will report that only 50 GB is in use, whereas the thin volume will often still show that 100 GB is consumed. If that user or application then goes on to write another 50 GB of data, the thin volume will report 150 GB consumed even though the filesystem will show only 100 GB used.

The reason for this is that most filesystems don't actually delete data when a user or application deletes data. Instead, they just mark the deleted data as no longer needed but leave the data in place. In this respect, the storage array has no idea that any data has been deleted, meaning that it can't reclaim and reuse any of the space that has been freed up in the filesystem. Also, most filesystems will write new data to free areas and will overwrite deleted data only if there is no free space left in the filesystem. This behavior is known as *walking forward* in the filesystem and causes TP-aware storage arrays to allocate new extents to TP volumes.

Fortunately, most storage arrays and most modern filesystems, such as the latest versions of NTFS and ext, are thin provisioning aware and implement standards such as T10 UNMAP so that storage arrays can be made aware of filesystem delete operations and free up the deleted space within the storage array.

An important concept to understand when it comes to thin provisioning is extent size. *Extent size* is the unit of growth applied to a thin volume as data is written to it.

For example, say a system has a TP extent size of 768 KB. Each time a host writes any data to an unused area of the TP volume, space will be allocated to the TP volume in multiples of 768 KB. The same would be true if the extent size was 42 MB; any time data is written to a new area of the TP volume, a 42 MB extent would be allocated to the volume. If your host has written only 4 KB of data to the TP volume, a 42 MB extent might seem a bit like overkill, and certainly from a capacity perspective, very large extent sizes are less efficient than small extent sizes. However, smaller extent sizes can have a performance overhead associated with them, because the system has to perform more extent allocations as data is written to a volume. Say a host writes 8 MB of data to a new TP volume. If the extent size was 42 MB, the system would have to allocate only a single extent, whereas if the extent size were 768 KB, it would have to allocate 11 extents, which is potentially 11 times as much work. In most situations, the capacity-related benefits of smaller extent sizes are seen as being more significant than the potential performance overhead of working with smaller extent sizes, because in the real world any potential performance overhead is so small that it's rarely, if ever, noticed.

Sometimes you might hear TP extents referred to as *pages*.

Real World Scenario

Make Sure You Test before Making Big Changes

A large retail company was in the process of deploying a shiny new storage array that sported TP. As part of this deployment, the company was migrating hosts off several older, low-end storage arrays and onto the new TP-aware array. Partway through the migrations, it was noticed that database backup and refresh jobs seemed to be taking longer on the shiny new array. At first glance, this should not have been the case. The new array had faster processors, more cache, more disks, and wide-striped TP volumes. After conclusive proof that backup and refresh jobs were taking longer on the new storage array than on the old storage arrays, the vendor was called in to explain why.

The root cause turned out to be the random way in which TP extents were allocated. Because TP extents were allocated on demand to the TP volumes, they were randomly scattered over all the spindles on the backend. This random layout negatively impacted the performance of the sequential backup and refresh operations. Extensive testing was done against multiple workload types that proved (on this particular implementation of TP) that TP volumes did not perform as well as traditional thick volumes for heavily sequential workloads.

A workaround was then put in place whereby all TP volumes that would be used for backup and refresh operations would be preallocated. With this option, available when creating new TP volumes, all extents are determined up front and are sequential, although still not allocated until data is written to them. This preallocation is slightly different than using thick volumes, because although all the extents that will make up the preallocated TP volumes are reserved up front, they are still not allocated until data is written to them. However, this workaround fixed the issue. The issue could also have been fixed by creating the backup and refresh volumes as thick volumes.

Overprovisioning

Most thin provisioning–based storage arrays allow an administrator to *overprovision* the array. Let's take a look at an example that explains how overprovisioning works and what its pros and cons are.

Let's start with a thin provisioning storage array that has 100 TB of physical capacity (after RAID overhead and so on) and 50 attached hosts. Now let's assume that the total amount of storage capacity provisioned to these 50 hosts totals 150 TB. Obviously, these volumes have to be thin provisioned. If they weren't, the maximum amount of capacity that could be provisioned from the array would be 100 TB. But because we've provisioned out 150 TB of capacity, and the array has only 100 TB of real physical capacity installed, we have overprovisioned our array. In overprovisioning terms, this array is 50 percent overprovisioned, or 150 percent provisioned.

NOTE

Most thin provisioning arrays—arrays that support thin provisioning and the ability to create thinly provisioned volumes and to be overprovisioned—also support the capability to create traditional thick volumes. So an array can often have a combination of thick and thin volumes. Often these arrays allow you to convert between thick and thin online without impacting or causing downtime to the host the volumes are presented to.

There are some obvious benefits to overprovisioning, but there are some drawbacks, too.

Overprovisioning allows an array to pretend it has more capacity than it does. This allows improved efficiency in capacity utilization. In our previous example, the array had 100 TB of physical usable storage and was overprovisioned by 50 percent. This array could still be heavily underutilized. It is entirely possible that although having provisioned out 150 TB to connected hosts, these hosts may have written only 60 TB to 70 TB of data to the array.

Now, let's assume the $/TB cost for storage on that array was $2,000/TB, meaning the array cost $200,000. If we had thickly provisioned the array, it would have been able to provision only 100 TB of capacity. In order to cater for the additional 50 TB requested, we would have had to buy at least 50 TB more capacity, at a cost of $100,000. However, because the array was thinly provisioned, we were able to overprovision it for zero cost to handle the additional requirement of 50 TB. So there are clear financial benefits to overprovisioning.

However, on the downside, overprovisioning introduces the risk that your array will run out of capacity and not be able to satisfy the storage demands of attached hosts. For example, say our array has only 100 TB of physical usable storage, and we've provisioned out 150 TB. If the attached hosts demand over 100 TB of storage, the array will not be able to satisfy that. As a result, hosts will not be able to write to new areas of volumes, which may cause hosts and applications to crash. You don't want this scenario to happen, and avoiding it requires careful planning and foresight. Overprovisioning should be done only if you have the tools and historical numbers available to be able to trend your capacity usage. You should also be aware of the kind of lead time required for you to be able to get a purchase order approved and your vendor to supply additional capacity. If it takes a long time for your company to sign off on purchases and it also takes your vendor a long time to ship new capacity, you will need to factor that into your overprovisioning policy.

Because of the risks just mentioned, it's imperative that the people ultimately responsible for the IT service be aware of and agree to the risks. That said, if managed well, overprovisioning can be useful and is becoming more and more common in the real world.

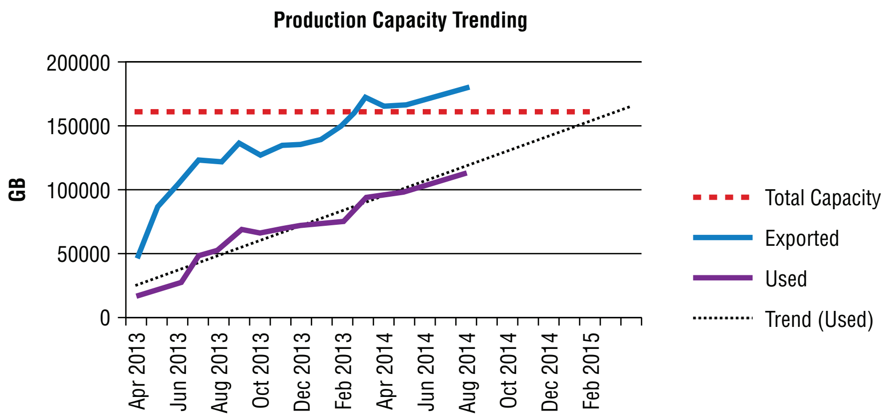
Trending

As I've mentioned the importance of trending in an environment that overprovisions arrays, let's take a moment to look at some of the metrics that are important to trend. This list is by no means exhaustive, but it covers metrics that are commonly trended by storage administrators who overprovision:

- Physical usable capacity
- Provisioned/allocated capacity
- Used capacity

As part of your trending, you will want to be tracking these metrics at least monthly. Figure 10.1 shows an example chart that has tracked the metrics for the past 16 months and has added a trend line predicting how used capacity is expected to increase over the next 6 months.

FIGURE 10.1 Overprovisioning tracking chart



As you can see, the array in Figure 10.1 will break the bank (used capacity will exceed physical usable capacity) somewhere around February 2015. Of course, you will want additional capacity to be installed into the array by November 2014. Graphs and trending charts like these are vital in all overprovisioning environments. Share them with management on a regular basis. Print them out, laminate them, and pin them on your desk so you don't ignore them!

Trigger Points for Purchase Orders

If you plan to overprovision in your environment, you will need to identify trigger points for new capacity purchases. These trigger points are conditions that cause you to purchase new capacity.

A common trigger point is based on *used capacity*. For example, you may have a trigger point that states that as soon as used capacity reaches 90 percent of physical usable capacity, you start ordering more storage. If you were using 90 percent capacity as a trigger point and you had an array with 200 TB of physical usable storage, you would start the process for installing new capacity as soon as used capacity reaches 180 TB.

It is vital that you don't wait until it's too late to get more capacity. The process of installing new capacity can be quite long in some organizations. You might include obtaining formal quotes, raising a purchase order (PO), following a PO sign-off process, placing the order and PO with the vendor, waiting for parts delivery, waiting for parts installation, and so on.

Following are three of the most common trigger points used by most organizations:

- Available unused capacity
- Percentage of available capacity
- Percentage overprovisioned

Determining Trigger Points for Capacity Purchases

In this step-by-step guide, you will walk through a scenario of deciding a trigger point for capacity purchases.

1. You deploy a storage array with 200 TB of usable storage and provision all volumes as thin provisioned volumes (TP volumes).
2. After six months of using the array, you have provisioned 150 TB of storage to your attached hosts and can see that only 50 percent of the storage provisioned is being used (150 TB provisioned, 75 TB used).
3. The array is now in steady state, meaning that you are no longer migrating lots of new hosts to it, so you expect its growth from now on to be steady.
4. After a further three months of use, you have provisioned 180 TB, but only 95 TB is used (still close to 50 percent used).

   At the current rate, you are provisioning an average of 10 TB per month and will therefore need a new storage array in two months. However, instead of purchasing a new storage array, you decide it would be a better idea to overprovision the array. After all, it still has 85 TB of capacity that is sitting there unused.
5. At this point, you decide to get buy-in and sign-off from senior management to enable overprovisioning. You provide the relevant statistics and graphs that show the growth rates and free space (physical usable capacity that is not currently used). As part of the sign-off, you agree with management that they will sign off on a new storage capacity purchase when you cross the 25 TB free space barrier. This figure of 25 TB is decided upon because you know that your current steady-state run rate is an increase of about 5 TB *used space* per month, and it takes the company up to six weeks to approve large IT infrastructure spending. It also takes a further two weeks for your storage vendor to ship additional storage. You also explain to management that when you get lower and lower on free space, the impact of an abnormal monthly increase in the amount of used storage could be severe. For example, if you have 50 TB of unused space available and a developer decides to consume 15 TB of unexpected space, you can roll with that, but if you have only 10 TB of unused available storage and the same developer decides to consume 15 TB, then all hell will break loose.
6. After management has agreed to overprovisioning the array, you make the necessary configuration changes to the array and keep monitoring and trending utilization.
7. After another six months, you have provisioned 225 TB from your array that has only 180 TB of physical usable storage, making your array 25 percent overprovisioned. Your used capacity is also now at 125 TB.

8. You can now see that over the past nine months, your used capacity has increased by about 5.5 TB per month. At this rate, you will run out of usable physical space in 12 to 13 months, so you still have plenty of growth left in the array.
9. After another nine months, your array looks like the following: used space is now at 175 TB, meaning you have only 25 TB of usable space left, and critically you have only about four to five months of growth left in the array before you run out of space and lose your job. Because of this, you push the button on a new capacity purchase for your array, and because you don't want to have to go and beg for more money for your budget this year, you buy enough capacity to last a little more than another 12 months (maybe something like 96 TB usable).

There are several reasons for choosing different trigger metrics or even utilizing more than one trigger. For example, 20 TB of free space might work well for a medium-sized array that has up to 200 TB of physical usable storage. 20 TB is 10 percent of a 200 TB array. However, if your array has 800 TB of usable physical storage, 20 TB is all of a sudden a mere 2.5 percent of total array capacity. And larger arrays tend to have more hosts attached and therefore grow faster than smaller arrays, meaning that 20 TB might last a while on a small array with a handful off hosts attached, but it won't last as long on a large array with hundreds of hosts attached.

Space Reclamation

Despite the huge benefits of thin provisioning, over time thin volumes get less and less thin, which is called *bloat*. Without a way to remove the bloat, you'd eventually get to a position where your thin volumes used up all of their possible space, making them little better than thick volumes. This is where space reclamation comes to the rescue, by giving us a mechanism to keep thin volumes thin.

One of the main reasons for bloat is that most traditional filesystems don't understand thin provisioning and don't inform storage arrays when they delete data. Let's look at a quick example. Assume a 100 GB filesystem on a 100 GB thin provisioned volume. If the host writes 90 GB of data to the filesystem, the array sees this and allocates 90 GB of data to the volume. If the host then deletes 80 GB of data, the filesystem will show 10 GB of used data and 90 GB of available space. However, the array has no knowledge that the host has just deleted 80 GB of data, and will still show the thin volume as 90 percent used. This bloat is far from ideal.

Now let's look at how space reclaim helps avoid this scenario.

Zero Space Reclamation

Space reclamation—often referred to as *zero space reclamation* or *zero page reclamation*—is a technology that recognizes deleted data and can release the extents that host the deleted data back to the free pool on the array, where they can be used for space for other volumes.

NOTE Traditionally, space reclamation technology has required deleted data to be zeroed out—overwritten by binary zeros. The array could then scan for contiguous areas of binary zeros and determine that these areas represented deleted data. However, more-modern approaches use technologies such as T10 UNMAP to allow a host/filesystem to send signals to the array telling it which extents can be reclaimed.

Traditional Space Reclamation

In the past, operating systems and filesystems were lazy when it came to deleting data. Instead of zeroing out the space where the deleted data used to be, the filesystem just marked the area as deleted, but left the data there. That's good from the operating system's perspective because it's less work than actually zeroing out data and it makes recovering deleted data a whole lot easier. However, it's no use to a thin provisioning storage array. As far as the array knows, the data is still there. In these situations, special tools or scripts are required that write binary zeros to the disk so that the array can then recognize the space as deleted and reclaim it for use elsewhere.

Once binary zeros were written to the deleted areas of the filesystem, the array would either dynamically recognize these zeros and free up the space within the array, or you would need an administrator to manually run a space reclamation job on the array. Both approaches result in the same outcome—increased free space on the array—but one required administrative intervention and the other didn't.

Reclaiming zero space on a host isn't too difficult. Exercise 10.1 walks you through how to do this.

EXERCISE 10.1

Reclaiming Zero Space on Non-TP-Aware Hosts and Filesystems

Let's assume we have a 250 GB TP volume presented to the host and formatted with a single 250 GB filesystem. Over time, the host writes 220 GB of data to the filesystem but deletes 60 GB, leaving the filesystem with 160 GB of used space and 90 GB of free space. However, as the filesystem is not TP aware, this 60 GB of deleted data is not recognized by the TP-aware storage array. In order for the storage array to recognize and free up the space, you need to perform the following steps:

1. Fill the 90 GB of free filesystem space with binary zeros. You may want to leave a few gigabytes free while you do this, just in case the host tries to write data at the same time you fill the filesystem with zeros. For all intents and purposes, the filesystem treats the zeros as normal data, so if the filesystem is filled with zeros, it behaves exactly the same as if it was filled with normal user data; it fills up and will not allow more data to be written to it.

2. After writing zeros to the filesystem, immediately delete the files that you created full of zeros so that your filesystem has space for new user writes.

3. On your storage array, kick off a manual zero space reclaim operation. This operation scans the volume for groups of contiguous zeros that match the size of the array's TP extent size. For example, if the array's TP extent size is 8 MB, the array will need to find 8 MB worth of contiguous zeros. Each time it finds a TP extent that is 100 percent zeros, it will release the TP extent from the TP volume and make it available to other volumes in the system.

An entire TP extent has to be full of zeros before it can be released from a TP volume and made available to other volumes. So if your TP extent size is 1 GB, you will need 1 GB worth of contiguous zeros before you can free an extent. If your extent size is 768 KB, you will need only 768 KB of contiguous zeros to free up an extent.

The UNMAP SCSI Standard

If all of this seems like a lot of work to free up space, you're right, especially in a large environment with thousands of hosts. Fortunately, a much slicker approach exists. Most modern operating systems include filesystems that are designed to work in conjunction with array-based thin provisioning.

T10 (a technical committee within the INCITS standards body responsible for SCSI standards) has developed a SCSI standard referred to as UNMAP, which is designed specifically to allow hosts and storage arrays to communicate information related to thin provisioning with each other. For example, when a host operating system deletes data from a filesystem, it issues SCSI UNMAP commands to the array, informing it of the extents that have been deleted. The array can then act on this data and release those extents from the volume and place them back in the array-free pool where they can be used by other hosts. This is way easier than having to manually run scripts to write zeros!

Fortunately, most modern operating systems, such as RHEL 6, Windows 2012, and ESXi 5.x, ship with thin provisioning–friendly filesystems and implement T10 UNMAP.

WARNING

Under certain circumstances, T10 UNMAP can cause performance degradation on hosts. You should test performance with T10 UNMAP enabled and also with it disabled. For example, the ext4 filesystem allows you to mount a filesystem in either mode—T10 UNMAP enabled or T10 UNMAP disabled. Following your testing, you should decide whether there will be instances in your estate where you want T10 UNMAP disabled.

Inline Space Reclamation

As previously mentioned, some storage arrays are able to watch for incoming streams of binary zeros and do either of the following:

- Not write them to disk if they are targeted at new extents in the volume (the host may be performing a full format on the volume)
- Release the extents they are targeted to if the extents are already allocated (the host is running a zero space reclaim script)

This is a form of inline space reclamation. It can usually be enabled and disabled only by a storage administrator because it has the potential to impact performance.

Older arrays tend to require an administrator to manually run space reclamation jobs to free up space that has been zeroed out on the host. This is a form of post-process reclamation. Although both approaches work, the inline approach is far simpler from an administration perspective.

Thin Provisioning Concerns

As with most things in life, there's no perfect solution that works for everything, and thin provisioning is no different. Although it brings a lot of benefits to the table, it also has one or two things you need to be careful about:

- Heavily sequential read workloads can suffer poor performance compared to thick volumes.
- There is the potential for a small write performance overhead each time new space is allocated to a thin volume.

As I keep mentioning, if you do plan to overprovision, make sure you keep your trending and forecasting charts up-to-date. Otherwise, you risk running out of space and causing havoc across your estate.

Be warned: overprovisioning does *not* simplify capacity management or storage administration. It makes it harder, and it adds pressure on you as a storage administrator. To gain the potential benefits, thin provisioning and overprovisioning require some effort.

Compression

Compression is a capacity optimization technology that enables you to store more data by using less storage capacity. For example, a 10 MB PowerPoint presentation may compress down to 5 MB, allowing you to store that 10 MB presentation using only 5 MB of disk space. Compression also allows you to transmit data over networks quicker while consuming less bandwidth. All in all, it is a good capacity optimization technology.

Compression is often file based, and many people are familiar with technologies such as WinZip, which allows you to compress and decompress individual files and folders and is especially useful for transferring large files over communications networks. Other types of compression include filesystem compression, storage array–based compression, and backup compression. Filesystem compression works by compressing all files written to a filesystem. Storage array–based compression works by compressing all data written to volumes in the storage array. And backup compression works by compressing data that is being backed up or written to tape.

There are two major types of compression:

- Lossless compression
- Lossy compression

As each name suggests, no information is lost with lossless compression, whereas some information is lost with lossy compression. You may well be wondering how lossy compression could be useful if it loses information. One popular form of lossy compression is JPEG image compression. Most people are familiar with JPEG images, yet none of us consider them poor-quality images, despite the fact that JPEG images have been compressed with a lossy compression algorithm. Lossy compression technologies like these work by removing detail that will not affect the overall quality of the data. (In the case of JPEGs, this is the overall quality of the image.) It is common for the human eye not to be able to recognize the difference between a compressed and noncompressed image file, even though the compressed image has lost a lot of the detail. The human eye and brain just don't notice the difference. Of course, lossy compression isn't ideal for everything. For example, you certainly wouldn't want to lose any zeros before the decimal point on your bank balance—unless, of course, your balance happens to be in the red!

How Compression Works

Compression works by reducing the number of bytes in a file or data stream, and it's able to do this because most files and data streams repeat the same data patterns over and over again. Compression works by re-encoding data so that the resulting data uses fewer bits than the source data.

A simple example is a graphic that has a large, blue square in the corner. In the file, this area of blue could be defined as follows:

```
Vector 0,0 = blue
Vector 0,1 = blue
Vector 0,2 = blue
Vector 0,3 = blue
Vector 0,4 = blue
Vector 0,5 = blue
Vector 0,6 = blue
Vector 0,7 = blue
…
Vector 53,99 = blue
```

A simple compression algorithm would take this repeating pattern and change it to something like the following:

```
vectors 0,0 through 53,99 = blue
```

This is much shorter but tells us the same thing without losing data (thus being lossless). All we have done is identified a pattern and re-encoded that pattern into something else that tells us exactly the same thing. We haven't lost anything in our compression, but the file takes up fewer bits and bytes.

Of course, this is only an example to illustrate the point. In reality, compression algorithms are far more advanced than this.

NOTE Compressing data that is already compressed does not typically result in smaller output. In fact, compressing data that is already compressed can sometimes result in slightly larger output. Audio, video, and other rich media files are usually natively compressed with content-aware compression technologies. So trying to compress an MP3, MP4, JPEG, or similar file won't usually yield worthwhile results.

Compression of primary storage hasn't really taken the world by storm. It exists and has some use cases, but its uptake has been slow to say the least.

NOTE *Primary storage* is used for data that is in active use. Examples of primary storage include DAS, SAN, and NAS storage that service live applications, databases, and file servers. Examples of *nonprimary storage*—often referred to as *near-line storage* or *offline* storage—include backup and archive storage.

Performance is the main reason that compression of primary storage hasn't taken the world by storm. Some compression technologies have the unwanted side-effect of reducing storage performance. However, if done properly, compression can improve performance on both spinning disk and solid-state media. Performance is important in the world of primary storage, so it's important to be careful before deploying compression technologies in your estate. Make sure you test it, and be sure to involve database administrators, Windows and Linux administrators, and maybe even developers.

Beyond primary storage, compression is extremely popular in the backup and archive space.

There are two common approaches to array-based compression (for primary storage as well as backup and archive):

- Inline
- Post-process

Let's take a look at both.

Inline Compression

Inline compression compresses data while in cache, before it gets sent to disk. This can consume CPU and cache resources, and if you happen to be experiencing particularly high I/O, the acts of compressing and decompressing can increase I/O latency. Again, try before you buy, and test, test, test!

On the positive side, inline compression reduces the amount of data being written to the backend, meaning that less capacity is required, both on the internal bus as well as on the drives on the backend. The amount of bandwidth required on the internal bus of a storage array that performs inline compression can be significantly less than on a system that does not perform inline compression. This is because data is compressed in cache before being sent to the drives on the backend over the internal bus.

Post-process Compression

In the world of post-process compression, data has to be written to disk in its uncompressed form, and then compressed at a later date. This avoids any potential negative performance impact of the compression operation, but it also avoids any potential positive performance impact. In addition, the major drawback to post-process compression is that it demands that you have enough storage capacity to land the data in its uncompressed form, where it remains until it is compressed at a later date.

Regardless of whether compression is done inline or post process, decompression will obviously have to be done inline in real time.

Performance Concerns

There's no escaping it: performance is important in primary storage systems. A major performance consideration relating to compression and primary storage is the time it takes to decompress data that has already been compressed. For example, when a host issues a read request for some data, if that data is already in cache, the data is served from cache in what is known as a *cache hit*. Cache hits are extremely fast. However, if the data being read isn't already in cache, it has to be read from disk. This is known as a *cache miss*. Cache misses are slow, especially on spinning disk–based systems.

Now let's throw decompression into the mix. Every time a cache miss occurs, not only do we have to request the data from disk on the backend, we also have to decompress the data. Depending on your compression algorithm, decompressing data can add a significant delay to the process. Because of this, many storage technologies make a trade-off between capacity and performance and use compression technologies based on Lempel-Ziv (LZ). LZ compression doesn't give the best compression rate in the world, but it has relatively low overhead and can decompress data fast!

The Future of Compression

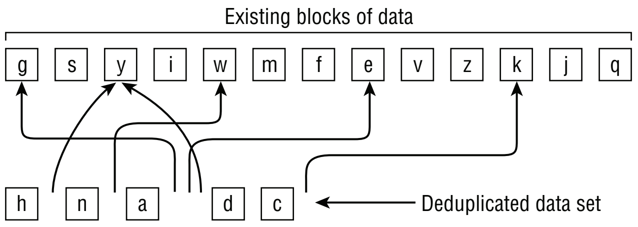
Compression has an increasingly bright future in the arena of primary storage. Modern CPUs now ship with native compression features and enough free cycles to deal with tasks such as compression. Also, solid-state technologies such as flash memory are driving adoption of compression. After all, compression allows more data to be stored on flash memory, and flash memory helps avoid some of the performance impact imposed by compression technologies.

Although the future is bright for compression in primary storage, remember to test before you deploy into production. Compression changes the way things work, and you need to be aware of the impact of those changes before deploying.

Deduplication

If you boil *deduplication* down to its most basic elements, you can see that it is all about identifying duplicate data and making sure it is stored only once. Storage systems that implement deduplication technologies achieve this by inspecting data and checking whether copies of the data already exist in the system. If a copy of this data already exists, instead of storing additional copies, pointers are used to point to the copy of the data. This is depicted in Figure 10.2.

FIGURE 10.2 Replacing duplicate data with pointers



At the top of Figure 10.2 is an existing pool of stored data. Below this pool is a new data set that has been written to the system. This data set is deduplicated, as any block in the data set that already exists in our pool of existing data is stripped out of the new data set and replaced with a pointer.

When replacing multiple copies of the same data with a single copy plus a pointer for each additional copy, the magic is that pointers consume virtually no space, especially when compared to the data they are replacing. If each square in our example in Figure 10.2 is 16 KB in size, and each pointer is 1 KB, we have saved 90 KB. Each of the six pointers consumes only 1 KB, giving a total of 6 KB to store the pointers, instead of the 6 × 16 KB that would have been required to store the six blocks of data.

Deduplication is similar to but subtly different from compression. Deduplication identifies data patterns that have been seen before, but unlike compression, these patterns don't have to be close to each other in a file or data set. Deduplication works across files and directories and can even work across entire systems that are petabytes (PB) in size. A quick and simple example follows. In the manuscript for this chapter, the word *storage* has been used 204 times (this may change by the time the chapter goes to production). A simple deduplication algorithm could take one instance of the word *storage* and remove every other instance, replacing them with a pointer back to the original instance. For this example, let's assume that the word *storage* consumes 4 KB, while a pointer consumes only 1 KB. If we don't deduplicate, and we stick with the 204 instances of the word, we'll consume 816 KB (204 × 4 KB). But if we do deduplicate, the 204 references to the word will consume only 207 KB (4 KB for the initial copy, plus 204 × 1 KB for the other 203 pointers). Now let's assume that our deduplication worked across all chapters of the book (each chapter is a separate Microsoft Word document); we would save even more space! Obviously this is an oversimplified example, but it works for getting the concept across.

Deduplication is not too dissimilar to compression. It has been widely adopted for backup and archive use cases, and has historically seen slow uptake with primary storage systems based on spinning disk. However, many of the all-flash storage arrays now on the market implement data deduplication.

NOTE It will serve you well to know that data that has been deduplicated can also be compressed. However, data that has been compressed cannot normally then be deduplicated. This can be important when designing a solution. For example, if you compress your database backups and then store them on a deduplicating backup appliance, you should not expect to see the compressed backups achieve good deduplication rates.

Block-Based Deduplication

Block-based deduplication aims to store only unique blocks in a system. And while the granularity at which the deduplication occurs varies, smaller granularities are usually better—at least from a capacity optimization perspective.

Two major categories of granularity are often referred to in relation to deduplication technologies:

- File level
- Block level

File-level granularity, sometimes referred to as *single-instance storage (SIS)*, looks for exact duplicates at the file level and shouldn't really be called deduplication.

Block-level deduplication works at a far smaller granularity than whole files and is sometimes referred to as a *subfile* by the SNIA. And because it works at a much finer granularity than files, it can yield far greater deduplication ratios and efficiencies.

File-level deduplication (single instancing) is very much a poor-man's deduplication technology, and you should be aware of exactly what you are buying! On the other hand, block-level deduplication with block granularity of about 4 KB is generally thought of as being up there with the best.

NOTE Deduplication is worked out as *data before ÷ data after,* and is expressed as a ratio. For example, if our data set is 10 GB before deduplication and is reduced to 1 GB after deduplication, our deduplication ratio is worked out as 10 ÷ 1 = 10, expressed as 10:1. This means that the original data set is 10 times the size of the deduplicated/stored data. Or put in other terms, we have achieved 90 percent space reduction.

Let's take a closer look at file single instancing and block-level deduplication.

File-Level Single Instancing

To be honest, file-level deduplication, also known as single instancing, is quite a crude form of deduplication. In order for two files to be considered duplicates, they must be *exactly the same*. For example, let's assume the Microsoft Word document for this chapter is 1.5 MB. If I save another copy of the file, but remove the period from the end of this sentence in that copy, the two files will no longer be exact duplicates, requiring both files to be independently saved to disk, consuming 3 MB. However, if the period were not removed, the files would be identical and could be entirely deduplicated, meaning only a single copy would be stored to disk, consuming only 1.5 MB.

That example isn't very drastic. But what happens when I create a PowerPoint file on Monday that is 10 MB and then on Tuesday I change a single slide and save it in a different folder? I will then have two files that will not deduplicate and that consume a total of about 20 MB, even though the two files are identical other than a minor change to one slide. Now assume I edit another slide in the same presentation on Wednesday and save it to yet another folder. Now I have three *almost-identical* files, but all stored separately and independently, consuming about 30 MB of storage. This can quickly get out of control and highlights the major weakness with file single instancing. It works, but it's crude and not very efficient.

Fixed-Length Block Deduplication

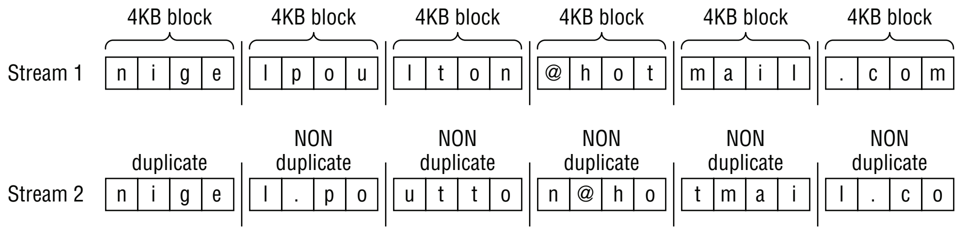
Deduplicating at the block level is a far better approach than deduplicating at the file level. Block-level deduplication is more granular than file-level deduplication, resulting in far better results.

The simplest form of block-level deduplication is *fixed block*, sometimes referred to as *fixed-length segment*. Fixed-block deduplication takes a predetermined block size—say, 16 KB—and chops up a stream of data into segments of that block size (16 KB in our example). It then inspects these 16 KB segments and checks whether it already has copies of them.

Let's put this into practice. We'll stick with a fixed block size of 16 KB, and use the 1.5 MB Word manuscript that I'm using to write this chapter. The 1.5 MB Word document gives us ninety-six 16 KB blocks. Now let's save two *almost-identical* copies of the manuscript. The only difference between the two copies is that I have removed the period from the end of this sentence in one of the copies. And to keep things simple, let's assume that this missing period occurred in the sixty-fifth 16 KB block. With fixed-block deduplication, the first sixty-four 16 KB blocks (1,024 KB) would be exactly the same in both files, and as such would be deduplicated. So we would need to store only one copy of each of the sixty-four 16 KB blocks—saving us 1 MB of storage. However, the thirty-two remaining 16 KB blocks would each be different, meaning that we cannot deduplicate them (replace them with pointers). The net result is that we could save the two almost-identical copies of the manuscript file, using up 2 MB of storage space instead of 3 MB. This is great, but it can be better.

Now then, the reason that all of the blocks following the omission of the period would be recognized as unique rather than duplicates is based on the nature of fixed-block deduplication. Quite often with fixed-block approaches to deduplication, as soon as you change one block in a data set, all subsequent blocks become offset and no longer deduplicate. This is the major drawback to fixed-block deduplication, as shown in Figure 10.3.

FIGURE 10.3 Fixed-length deduplication



The example in Figure 10.3 shows two almost-identical email addresses. The only difference is the period in the local part of the email address in stream 2. As you can see, every block following the inclusion of the period is offset by one place, and none of the following blocks are identical, meaning none can be deduplicated.

Variable Block Size

Variable block, also known as *variable length segment*, approaches to deduplication pick up where fixed block started to struggle. Variable-block approaches apply a dynamic, floating boundary when segmenting a data set, reducing, even further, the impact of making changes to blocks. They also base their decisions about where each block begins and ends not by size alone but also by recognizing repeating patterns in the data itself. This means when there's an insertion, or removal, as in our period example, the variable-block method will realign with the existing data when it sees the repeating patterns. Most of the time, variable block size deduplication technologies will yield greater storage efficiencies than both fixed block and file single instancing.

Hashing and Fingerprinting

Let's take a look at the most common way of indexing and determining duplication of blocks.

Every block of data that is sent to and stored in a deduplicating system is given a digital fingerprint (unique hash value). These fingerprints are indexed and stored so they can be looked up very quickly. They're often stored in cache. If the index is lost, it can usually be re-created, although the process of re-creating can be long and keep the system out of service while it's being re-created.

When new writes come into a system that deduplicates data, these new writes are segmented, and each segment is then fingerprinted. The fingerprint is compared to the index of known fingerprints in the system. If a match is found, the system knows that the incoming data is a duplicate (or potential duplicate) block and replaces the incoming block with a much smaller pointer that points to the copy of the block already stored in the system.

NOTE

Depending on the strength of the digital fingerprinting system, a system may choose to perform a *bit-for-bit compare* each time a duplicate block is identified. In a bit-for-bit compare, all bits in a block that is a potential duplicate are compared with the bits in the block already stored on the system. This comparison is performed to ensure that the data is indeed a duplicate and not what is known as a *collision*, in which two unique blocks of data produce the same fingerprint. Systems that employ weak fingerprinting need to perform more bit-for-bit compare operations because weak fingerprinting systems produce a higher rate of collisions. On the positive side, weak fingerprinting systems usually have a low performance overhead and are fine as long as you can live with the overhead of having to perform more bit-for-bit compares. All-flash storage arrays are one instance where the overhead of performing bit-for-bit compares is very low, because reading data from flash media is extremely fast.

Where to Deduplicate

Data can be deduplicated in various places. In typical backup use cases, deduplication is commonly done in one or more of the following three ways:

- Source
- Target
- Federated

Let's look at each option in turn.

Source-Based Deduplication

Source-based deduplication—sometimes referred to as *client-side deduplication*—requires agent software installed on the host and is where the heavy lifting work of deduplication is done on the host, consuming host resources including CPU and RAM. However, source-based deduplication can significantly reduce the amount of network bandwidth consumed between the source and target—store less, move less! This can be especially useful in backup scenarios where a large amount of data is being streamed across the network between source and target.

Because of its consumption of host-based resources and its limited pool of data to deduplicate against (it can only deduplicate against data that exists on the same host), it is often seen as a lightweight *first-pass deduplication* that is effective at reducing network bandwidth. Consequently, it is often combined with target-based deduplication for improved deduplication ratios.

Target-Based Deduplication

Target-based deduplication, sometimes referred to as *hardware-based deduplication*, has been popular for a long time in the backup world. In target-based deduplication, the process of deduplication occurs on the target machine, such as a deduplicating backup appliance. These appliances tend to be purpose-built appliances with their own CPU, RAM, and persistent storage (disk). This approach relieves the host (source) of the burden of deduplicating, but it does nothing to reduce network bandwidth consumption between source and target.

It is also common for deduplicating backup appliances to deduplicate or compress data being replicated between pairs of deduplicating backup appliances. There are a lot of deduplicating backup appliances on the market from a wide range of vendors.

Federated Deduplication

Federated deduplication is a distributed approach to the process of deduplicating data. It is effectively a combination of source and target based, giving you the best of both worlds—the best deduplication ratios as well as reduced network bandwidth between source and target.

This federated approach, whereby the source and target both perform deduplication, can improve performance by reducing network bandwidth between the source and target, as well as potentially improving the throughput capability of the target deduplication appliance. The latter is achieved by the data effectively being pre-deduplicated at the source before it reaches the target.

Technologies such as Symantec OpenStorage (OST) allow for tight integration between backup software and backup target devices, including deduplication at the source and target.

It is now common for many backup solutions to offer fully federated approaches to deduplication, where deduplication occurs at both the source and target.

When to Deduplicate

I've already mentioned that the process of deduplication can occur at the source, or target, or both. But with respect to target-based deduplication, the deduplication process can be either of the following:

- Inline
- Post-process

Let's take a look at both approaches.

Inline Deduplication

In *inline deduplication*, data is deduplicated in real time—as it is ingested, and before being committing to disk on the backend. Inline deduplication often deduplicates data while in cache. We sometimes refer to this as *synchronous deduplication*.

Inline deduplication requires that all hashes (fingerprinting) and potential bit-for-bit compare operations must occur before the data is committed to disk. Lightweight hashes are simpler and quicker to perform but require more bit-for-bit compares, whereas stronger hashes are more computationally intensive but require fewer bit-for-bit compare operations.

The major benefit to inline deduplication is that it requires only enough storage space to store deduplicated data. There is no need for a large landing pad where non-deduplicated data arrives and is stored until it is deduplicated at a later time.

The potential downside to inline deduplication is that it can have a negative impact on data-ingestion rates—the rate, in MBps, that a device can process incoming data. If a system is having to segment, fingerprint, and check for duplications for all data coming into the system, there is a good chance this may negatively impact performance.

Although inline deduplication works really well for small, medium, and large businesses, they can be particularly beneficial for small businesses and small sites for which the following are true:

- You want to keep your hardware footprint to a minimum, meaning you want to deploy as little storage hardware as possible.
- Your backup window—the time window during which you can perform your backups—is not an issue, meaning that you have plenty of time during which you can perform your backups.

In these situations, inline deduplication can be a good fit. First, you need to deploy only enough storage required to hold your deduplicated data. Also, because you are not having to squeeze your backup jobs into a small time window, the potential impact on backup throughput incurred by the inline deduplication process is not a concern.

Post-process Deduplication

Post-process deduplication is performed asynchronously. Basically, data is written to disk without being deduplicated, and then at a later point in time, a deduplication process is run against the data in order to deduplicate it.

In a post-process model, all hashes and bit-for-bit compare operations occur after the data has been ingested, meaning that the process of deduplicating data does not have a direct impact on the ability of the device to ingest data (MBps throughput). However, on the downside, this approach requires more storage, as you need enough storage capacity to house the data in its non-deduplicated form, while it waits to be deduplicated at a later time.

Most backup-related deduplication appliances perform inline deduplication, because most modern deduplication appliances have large amounts of CPU, RAM, and network bandwidth to be able to deal relatively well with inline deduplication.

WARNING

Be careful with using post-process deduplicating technologies as backup solutions, because the background (post-process) deduplication operation may not complete before your next round of backups start. This is especially the case in large environments that have small backup windows in which they have to back up large quantities of data.

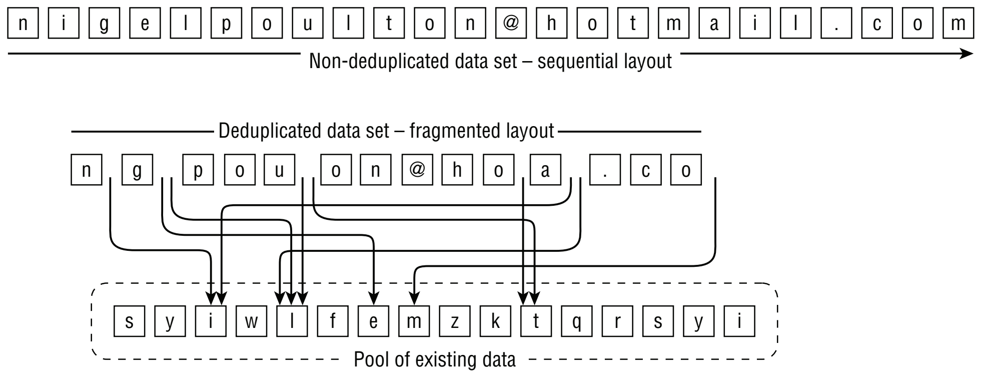
Backup Use Case

Deduplication technologies are now a mainstay of just about all good backup solutions. They are a natural fit for the types of data and repeated patterns of data that most backup schedules comprise. During a single month, most people back up the same files and data time and time again. For example, if your home directory gets a full backup four times each month, that is at least four times each month that the same files will be backed up, and that's not considering any incremental backups that might occur between the four full backups mentioned. Sending data like this to a deduplication appliance will normally result in excellent deduplication.

Deduplicating backup-related data also enables longer retention of that data because you get more usable capacity out of your storage. You might have 100 TB of physical storage in your deduplication appliance, but if you are getting a deduplication ratio of 5:1, then you effectively have 500 TB of usable capacity. Also, as a general rule, the longer you keep data (backup retention), the better your deduplication ratio will be. This is because you will likely be backing up more and more of the same data, enabling you to get better and better deduplication ratios.

However, deduplication technologies tend to fragment file layouts. Duplicated data is referenced by pointers that point to other locations on the backend. Figure 10.4 shows the potential sequential layout of a data set before being deduplicated, as well as the same data set after deduplication. Although the deduplicated version of the filesystem consumes far less space, reading back the entire contents of the deduplicated data set will be a lot more difficult and take a lot more time than reading back the entire contents of the non-deduplicated data set. This is especially the case with spinning disk–based systems, though it is much less of an issue for solid-state systems.

FIGURE 10.4 Fragmentation caused by deduplication


n i g e l p o u l t o n @ h o t m a i l . c o m
Non-deduplicated data set – sequential layout
Deduplicated data set – fragmented layout
n g p o u o n @ h o a . c o
s y i w l f e m z k t q r s y i
Pool of existing data

The fragmenting of data layout has a direct impact on backup-related technologies such as synthetic full backups. A *synthetic full backup* is a full backup that is not created the normal way by sending all data from the source to the target. Instead, a synthetic full backup is created by taking a previous full backup that already exists on the target and applying any incremental backups that have occurred since that full backup, effectively creating a new full backup without having to read all data on the source and send it across the network to the target. Although synthetic full backups are popular these days, they are a challenge for deduplication systems. In order to create the synthetic full backup, a previous full backup has to be read from the deduplication appliance, plus any incremental backups taken since, and at the same time the deduplication appliance has to write out the new synthetic full backup. All of this reading and writing (which includes lots of heavy random reading because the deduplicated data is fragmented on the system) can severely impact the performance of the deduplication appliance.

Virtualization Use Case

Deduplication can lend itself well to virtualization technologies such as server and desktop virtualization. These environments tend to get good deduplication ratios thanks to the fact that *x* number of virtual Windows servers will share a very high percentage of the same core files and blocks. The same goes for other operating systems such as Linux.

For example, if you have a 10 TB image of a Windows server and deploy 100 servers from that image, those 100 servers will share a massive number of identical files and blocks. These will obviously deduplicate very well. Deduplication like this can also lead to potential performance improvements if you have a large read cache, because many of these shared files and blocks will be highly referenced (frequently read) by all virtual machines, causing them to remain in read cache where they can be accessed extremely quickly. If each of these 100 virtual servers had to access its own core files in individual locations on disk in our storage array, this would require our storage array to do a lot of work seeking and reading from the disks on the backend. This would result in poor performance. However, if these core files and blocks were deduplicated and only the master copies were frequently accessed, they could be accessed from cache memory at much faster speeds than if each read had to go to the backend disks.

Primary Storage and Cloud Use Cases

The rise of all-flash arrays has started pushing deduplication technologies to the forefront in primary storage use cases. Traditionally, deduplication has been avoided for primary storage systems because of the performance impact. However, that performance impact was mainly due to the nature of spinning disk and its poor random-read performance. After all, good random-read performance is vital for the following two deduplication-related operations:

- Performing bit-for-bit compare operations
- Performing read operations on deduplicated data

If bit-for-bit compare operations take a long time, they will impact the overall performance of the systems. Also, because the process of deduplicating data often causes it to be fragmented, reading deduplicated data tends to require a lot of random reading. All-flash arrays don't suffer from random-read performance problems, which makes deduplication technologies in primary storage very doable. In fact, most all-flash arrays support inline deduplication, and it's so tightly integrated to some all-flash arrays that they don't allow you to turn the feature off!

Tier 3 and cloud are also places where deduplication technologies are a good fit because accessing data from lower-tier storage, and especially from the cloud, is accepted to be slow. Adding the overhead of deduplication on top of this is not a major factor on the performance front, but it can be significant at reducing capacity utilization.

Deduplication Summary

Deduplication is similar in some ways to compression, but it is not the same. That said, deduplication and compression can be used in combination for improved space savings, though you will need to know the following:

- Deduplicated data can be compressed.
- Compressed data usually cannot be deduplicated.

If you want to use the two in combination, dedupe first, and compress second.

Real World Scenario

Mixing Compression and Deduplication

A large banking organization had purchased several large deduplicating backup appliances to augment its existing backup and recovery process. The idea was to send backups to the deduplicating backup appliances before copying them to tape so that restore operations could be done more quickly. The solution became unstuck when database backups began being compressed before being sent to the deduplicating backup appliance. Originally, the database backups had been written to a SAN volume before being copied to the deduplicating backup appliance, but as is so often the case, as the databases grew, more storage was assigned to them. However, there wasn't enough money in the budget to expand the SAN volumes that the database backups were being written to. Instead, the decision was made to compress the database backups so that they would still fit on the SAN volumes that weren't increased in size.

The problem was that the organization was so large and disjointed that the database guys didn't talk to the backup guys about this decision, and they weren't themselves aware that compressed data doesn't deduplicate. After a while of running like this, where database backups could no longer be deduplicated by the deduplication appliance, it was noticed that the deduplication ratios on the deduplication appliances were dropping and that space was running short before it should have been.

This situation was resolved by deploying database backup agents on the database servers, allowing them to back up directly to the deduplicating backup appliance without needing to be written to an intermediary SAN volume and therefore without needing to be compressed.

Some data types will deduplicate better than others; that's just life.

Deduplicating data tends to lead to random/fragmented data layout, which has a negative performance impact on spinning disk drives, but it is not a problem for flash and other solid-state media.

File-level deduplication is a poor-man's deduplication and does not yield good deduplication ratios. Variable-block deduplication technologies that can work at small block sizes tend to be the best.

Auto-Tiering

Sub-LUN auto-tiering is available in pretty much all disk-based storage arrays that deploy multiple tiers of spinning disk, and sometimes a smattering of flash storage. While it doesn't allow you to increase your usable capacity as deduplication and compression technologies do, it absolutely does allow you to optimize your use of storage resources.

Auto-tiering helps to ensure that data sits on the right tier of storage. The intent usually is for frequently accessed data to sit on fast media such as flash memory or 15K drives, and infrequently accessed data to sit on slower, cheaper media such as high-capacity SATA and NL-SAS drives or maybe even out in the cloud.

The end goal is to use your expensive, high-performance resources for your most active, and hopefully most important, data, while keeping your less frequently accessed data on slower, cheaper media.

NOTE

If your storage array is an all-flash array, it will almost certainly not use auto-tiering technologies, unless it tiers data between different grades of flash memory, such as SLC, eMLC, and TLC. However, few all-flash arrays on the market do this, although some vendors are actively considering it.

Sub-LUN Auto-Tiering

Early implementations of auto-tiering worked at the volume level; tiering operations, such as moving up and down the tiers, acted on entire volumes. For example, if part of a volume was frequently accessed and warranted promotion to a higher tier of storage, the entire volume had to be moved up. This kind of approach could be extremely wasteful of expensive resources in the higher tiers if, for example, a volume was 200 GB and only 2 GB of it was frequently accessed and needed moving to tier, 1 but auto-tiering caused the entire 200 GB to be moved.

This is where sub-LUN auto-tiering technology came to the rescue. *Sub-LUN tiering* does away with the need to move entire volumes up and down through the tiers. It works by slicing and dicing volumes into multiple smaller extents. Then, when only part of a volume needs to move up or down a tier, only the extents associated with those areas of the volume need to move. Let's assume our 200 GB volume is broken into multiple extents, each of which is 42 MB (as implemented on most high-end Hitachi arrays). This time, if our 200 GB volume had 2 GB of hot data (frequently accessed) that would benefit from being moved to tier 1, the array would move only the 42 MB extents that were hot, and wouldn't have to also move the cold (infrequently accessed) 42 MB extents. That is a far better solution!

Obviously, a smaller extent size tends to be more efficient than a larger extent size, at least in the context of efficiently utilizing capacity in each of our tiers. Some mid-range arrays have large extent sizes, in the range of 1 GB. Moving such 1GB extents is better than having to move an entire volume, but if all you need to move up or down the tiers is 10 MB of a volume, it could still be improved on. In this case, an even smaller extent size, such as 42 MB or even smaller, should clearly be more efficient. Smaller extent sizes also require less system resources to move around the tiers. For example, moving 1 GB of data up or down a tier takes a lot more time and resources than moving a handful of 42 MB extents. However, small extent sizes have a trade-off too. They require more system resources, such as memory, to keep track of them.

Tiers of Storage

In the world of tiering, three seems to be the magic number. The vast majority of tiered storage configurations (maybe with the exception of purpose-built hybrid arrays) are built with three tiers of storage.

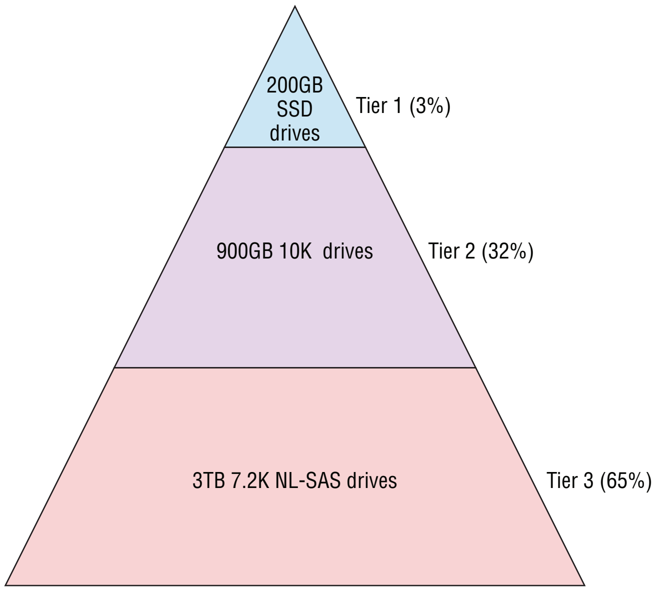
Exactly what type of storage you put in each tier depends, but a common approach is for tier 1 to be solid-state storage (flash memory), tier 2 to be fast spinning disk such as 10K or 15K SAS drives, and tier 3 to be big, slow SATA or NL-SAS drives. However, how much of each tier you have and similar concerns are highly dependent upon your environment and your workload. If possible, you should work closely with your vendors and technology partners to determine what is right for your environment.

A common three-tiered design used in the real world is shown in Figure 10.5. The percentage values represent how much of the overall system capacity is in each tier. They do not refer to the percentage of overall disk drives each tier has.

Activity Monitoring

To make tiering-related decisions such as which extents to move up the tiers and which to move down the tiers, a storage array needs to monitor activity on the array. This monitoring usually involves keeping track of which extents on the backend are accessed the most and which are accessed the least. However, the times of day when you monitor this activity can be crucial to getting the desired tiering results. For example, many organizations choose to monitor data only during core business hours, when the array is servicing line-of-business applications. By doing this, you are ensuring that the extents that move up to the highest tiers are the extents that your key line-of-business applications use. In contrast, if you monitor the system outside business hours (say, when your backups are running), you may end up with a situation where the array moves the extents relating to backup operations to the highest tiers. In most situations, you won't want backup-related volumes and extents occupying the expensive high-performance tiers of storage, while your line-of-business applications have to make do with cheaper, lower-performance tiers of storage.

FIGURE 10.5 Pyramid diagram of tier quantities



Arrays and systems that let you fine-tune monitoring periods are better than those that give you less control. The more you understand about your environment and the business that it supports, the better you will be able to tune factors such as the monitoring period, and the better the results of your tiering operations will be.

Tiering Schedules

Similar to fine-tuning and custom-scheduling monitoring windows, the same can be done for scheduling times when data can move between tiers. As with creating custom monitoring periods, defining periods when data can move between tiers should be customized to your business. After all, moving data up and down tiers consumes system resources, resources that you probably don't want consumed by tiering operations in the middle of the business day when your applications need those resources to service customer sales and the perform similar functions.

A common approach to scheduling tiering operations (moving data up and down through the tiers) is to have them occur outside business hours—for example, from 3 a.m. to 5 a.m.—when most of your systems are quiet or are being backed up.

Configuring Tiering Policies

As well as customizing monitoring and movement windows, it will often be useful to configure policies around tiering. Some common and powerful polices include the following:

- Exclusions
- Limits
- Priorities

Exclusions allow to you do things such as exclude certain volumes from all tiering operations, effectively pinning those volumes into a single tier. They also allow you to do things such as exclude certain volumes from occupying space in certain tiers. For example, you may not want volumes relating to backups to be able to occupy the highest tiers.

Limits allow you limit how much of a certain tier a particular volume or set of volumes can occupy.

Priorities allow you to prioritize certain volumes over others when there is contention for a particular tier. As an example, your key sale app might be contending with a research app for tier 1 flash storage. In this situation, you could assign a higher priority to volumes relating to your sales app, giving those volumes preference over other volumes and increasing the chances that the volumes associated with the sales app will get the space on tier 1.

Of course, not all arrays and systems support all features. Do your research before assuming your choice of technology can do all of the preceding tasks.

Real World Scenario

The Importance of Testing Performance and Deploying Changes in a Staged Approach

A large company that was an early pioneer of sub-LUN auto-tiering technology learned a lot of lessons the hard way, despite doing a lot of planning before deploying the technology.

One such lesson learned concerned the amount of tier 1 flash space a single volume could occupy. The company was partway through deploying their first array using sub-LUN auto-tiering when a major performance problem was encountered. They had been carefully migrating applications to the new storage array over several weekends. Then, on a Friday before another planned migration weekend, several important applications experienced a significant drop in performance. The performance drop was enough to postpone the planned migrations for the coming weekend. An investigation began, and to cut a long story short, the root cause was a recently added application that was so performance hungry that it had muscled its way into most of the available tier 1 flash space. But, in occupying so much of the tier 1 flash space, it had forced other important applications down to lower tiers.

This problem was resolved by restricting the amount of tier 1 flash storage that the newly migrated application could occupy.

In my experience, such fine-tuning is rarely needed, but this example highlights a potential danger with auto-tiering technologies: performance can vary from day to day, depending on how each application is moved around the tiers. If today you have 10 percent of your volumes in tier 1 flash, but tomorrow you have only 1 percent in tier 1 flash, there's a good chance you will experience lower performance tomorrow.

Array Size Concerns

Unfortunately, auto-tiering solutions don't suit every scenario. One scenario where they may not be the best choice is smaller arrays and smaller storage systems. A major reason for that is the number of spindles (drives) each tier of storage has. For example, a small storage array with 32 drives installed might be well suited to a single tier of storage configured into a single pool, meaning that all volumes have access to all 32 spindles. If you try to divide such a small system into three tiers of storage (for example, four flash drives, twelve 15K drives, and sixteen SATA drives), you may find that each tier has too few spindles to make any tier performant enough, and the benefits of the flash storage may be outweighed by the backend operations required to move data up and down the tiers. Also, auto-tiering is sometimes a licensed feature, adding cost to an array. Often this additional cost isn't beneficial on small arrays.

As a general rule, instead of utilizing auto-tiering on smaller arrays, you *may* be better off having a single tier of medium- to high-performance drives such as 10K SAS drives, maybe supplemented with a flash cache, where flash drives are utilized as a second level of cache rather than a persistent tier of storage.

Remote Replication

An often unconsidered side-effect of auto-tiering is the potential imbalance between source and target arrays that are replicated. For example, in a replicated configuration where one array is considered the primary array (has live hosts attached to it) and the other array considered the secondary array (has standby hosts attached to it), the primary array sees all of the read and write workload. As a result of this read and write workload, the auto-tiering algorithms of the array move extents up and down the tiers, ensuring an optimal layout. However, the target array does not see the same workload. All the target array sees is replicated I/O, and only write I/Os are replicated from the source to the target. Read I/Os are not replicated. As a result, how a volume is spread across the tiers on the target array will often be very different from how it's spread across the tiers on the source array. This frequently results in a volume being on lower-performance tiers on the target array. This can be a problem if you ever have to bring systems and applications up on the target array, because they may perform lower than they did on the source array. This is just one more thing to consider with auto-tiering.

Summary

In this chapter we covered all of the prevalent capacity optimization technologies in the modern storage world. We started out by talking about Thin Provisioning and the difference between thick and thin volumes, from both a capacity and performance perspective. We then talked about how thin provisioning enables overprovisioning, and the potential cost savings this can bring. We finished off our discussion on thin provisioning by talking about zero space reclamation technology and how this is maturing and helping us keep thin volumes thin. We then talked about inline and post-process compression deduplication, being careful to point out the differences and use cases for each, as well as what to watch out for when using the two technologies side-by-side. We finished the chapter talking about auto-tiering and how it enables us to more efficiently use our capacity by placing the right data on the right tiers of storage.

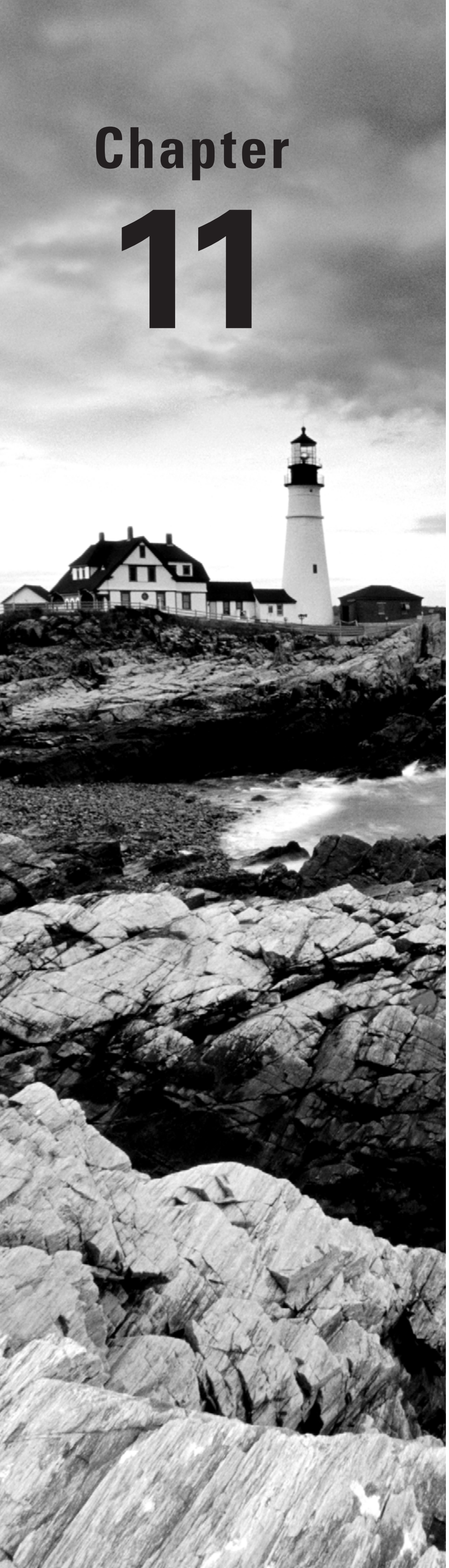
Chapter Essentials

Thin Provisioning Thin provisioning (TP) technologies allow you to loan out more storage than you physically have, resulting in higher utilization of storage resources. This is known as overprovisioning and must be managed and monitored so you do not overrun the physical capacity in your storage array.

Compression Compression technologies re-encode data so it can be stored and transmitted using fewer bits and bytes, enabling you to store more data with less physical capacity, as well as transmit data quicker over networks consuming less bandwidth. Compression technologies are part of all good backup solutions and are increasingly popular in all-flash storage arrays.

Deduplication Deduplication technologies allow you to squeeze more usable capacity out of your storage systems. Deduplication works by recognizing duplicate blocks of data, and replacing the duplicate blocks with pointers. Variable-block deduplication is recognized as a leading approach to deduplication. Deduplication technologies are part of all good backup solutions and are increasingly popular in all-flash storage arrays.

Sub-LUN Auto-Tiering While sub-LUN auto-tiering technologies do not increase your usable capacity, they do allow you to more efficiently utilize your storage resources. Sub-LUN auto-tiering works by placing frequently accessed blocks of data on high-performance media such as solid-state media, while placing infrequently accessed blocks of data on slower, cheaper storage media. Smaller extent sizes yield better efficiencies with sub-LUN auto-tiering technologies.



Chapter 11 Backup and Recovery

TOPICS COVERED IN THIS CHAPTER:

- ✓ Why we back up
- ✓ Backup methods
- ✓ Backup types
- ✓ Backup targets
- ✓ Backup to the cloud
- ✓ Backup retention
- ✓ Archiving

In this chapter, you'll learn about server and application backups and restores. Although you'll be focusing mainly on server and application backup and recovery, many of the same principles apply to backing up devices such as laptops and tablets.

The major lesson in this chapter is that backup is all about recovery. This point can't be stressed enough! Burn this fact into your head and make sure you never forget it. If you back up your data every day but can't restore data from those backups, your backups are useless and you're wasting your time.

You'll also learn about backup methods, such as hot backups and LAN-free backups, as well as the impact they have on both networks and application performance and recoverability. You'll learn about the types of backups, such as full backups, incremental backups, differential backups, and synthetic full backups, and how each impacts restore, application, and network performance. This chapter also covers backup devices such as tapes, tape drives, virtual tape libraries, and even other less-common backup media types such as Blu-ray Discs.

You'll see how capacity optimization technologies affect backup performance, cost, and potentially performance of the restore.

The chapter also covers archiving and how it differs from but also complements backup. And then there's the impact of ever-increasing legislation and regulatory audit requirements that govern so much of data archiving in the modern world. You'll learn about technologies that are great for archiving, such as content addressable storage and write once, read many technologies.

But again, the major lesson from this chapter is that backup is all about being able to recover lost or corrupt data—and recovering it in a timely and efficient, although not necessarily super fast, manner.

Why We Back Up

Backups aren't sexy. They cost money. They can be cumbersome to manage. And they can impact application performance. So why do we do them? Well, for starters, backups can save your data and potentially your job and your company—and that's no understatement. Basically, no data equals no business!

It's also no secret that data can be corrupted or accidentally deleted. When this happens, backups suddenly become the most important things in the world, and everyone sits around nervously chewing their fingernails in the hope that the data was recently backed up and can be restored.

The following are true statements that relate to the importance of backups:

- If the business can't go backward after a disaster, it can't go forward.
- If it's not backed up, it doesn't exist.
- If it doesn't exist in at least two places, it isn't backed up.

These statements form some of the fundamental principles of backup.

Now let's talk about the role of backup in disaster recovery and business continuity planning, as well as defining some terminology fundamental to backup and recovery.

Backups and Business Continuity

One of the major roles of backup and recovery is functioning as part of a *business continuity plan (BCP)*. In a BCP, backups sit alongside other storage-related technologies such as replication and snapshots, though we should note that replicating data is absolutely not the same as backing up, and snapshots in and of themselves do not constitute backups. Anyway, backups should form a part of just about every business continuity plan.

Recovery Point Objectives

A *recovery point objective (RPO)* is the point in time to which a service can be restored/recovered. So, if you have an RPO of 24 hours, you will be able to recover a service or application to the state it was in no more than 24 hours ago. For example, a backup schedule that backs up a database every day at 2 a.m. can recover that database to the state it was in at 2 a.m. every day. At any given point in time, you will have a valid backup of the database that is no more than 24 hours old—within your stated RPO of 24 hours.

RPOs form part of a wider service-level agreement (SLA) between IT and the business. In the context of IT, an SLA is a form of contract outlining the levels of service that IT will provide to a customer. The customer is usually the business, or a business unit, if you're an internal IT department. For example, an SLA might define that all data will be backed up daily, and be recoverable within 6 hours of a request. Almost all SLAs will contain recovery point objectives.

When it comes to backups, recovery point objectives are a major consideration. If the business requires an RPO of 5 minutes for a certain application, then a daily backup of that system will not meet that RPO, and you will need to use different technologies such as snapshots.

With traditional tape-based backups, the normal minimum RPO is one day (24 hours), because it's usually not realistic to make traditional backups to tape more frequently than once per day. This is for two major reasons:

- Many backup methods impact the performance of the system being backed up. So making frequent traditional backups to tape could negatively impact the performance of the system being backed up.
- Many corporate backup systems (media servers, tape drives, backup networks) are already very busy taking just a single backup per day.

In these situations, if more-granular RPOs are required, traditional backups can be augmented with technologies such as snapshots. These snapshots could be taken, for example, every 15 minutes but kept for only one day. Then each full traditional backup to tape could be taken once per day and kept for 12 months. This configuration will allow for a recovery of the application to any 15-minute point within the last 24 hours.

Interestingly, within the context of backups, there is also the concept of *retention*. For example, if you take daily backups of your corporate database server, keep those backups for 12 months, and then expire (purge) them as soon as they are older than 12 months, you will not be able to recover the corporate database system to a state it was in more than 12 months ago. In this example, the retention period for backups of the corporate database system will be 12 months (365 days). And while this might not seem like a problem, legislation and regulatory requirements might state that data from a given system must be available to restore from the last 25 years.

Recovery Time Objectives

A *recovery time objective (RTO)* is significantly different from a recovery point objective. Whereas a recovery point objective says that you can recover a system or application to how it was within a certain given time window, a recovery time objective states how long it will take you to do that. For example, it might take you 8 hours to recover a system to the state it was in 24 hours ago.

And when defining and deciding RTOs, make sure you factor in things like these:

- The time it takes to recall tapes from off-site storage
- The time it takes to rally staff and get logged on and working on the recovery

For example, if you have no staff working on a Saturday morning, and all tapes are off-site in a secure storage facility, you will need to factor both of those facts into the RTO to handle the possibility of a user logging a request at 6 a.m. on a Saturday morning to restore some data. It's not as simple as saying that you can restore *x* GB per minute, so to restore *y* GB will take *z* minutes. There are other significant factors to consider.

Exercise 11.1 shows you how to create a simple scheduled backup.

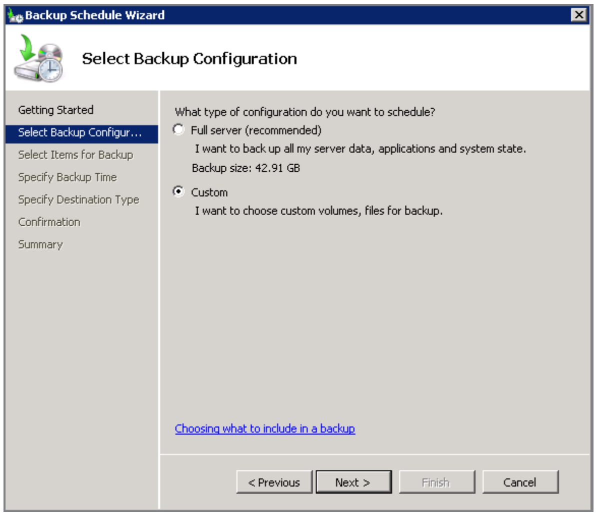
EXERCISE 11.1

Creating a Simple Scheduled Backup

This step-by-step activity will walk you through the process of creating an hourly scheduled backup of a research folder to a SAN disk that is mounted locally as the F: drive on a Windows server. The schedule will take a backup of the `c:\research folder` of the server every hour between 7 a.m. and 7 p.m.

1. From the Windows desktop, click Start ➪ Administrative Tools ➪ Windows Server Backup.
2. From within the Windows Server Backup tool, choose Action ➪ Backup Schedule.
3. On the Getting Started screen, click Next.

4. On the Select Backup Configuration screen, select Custom.

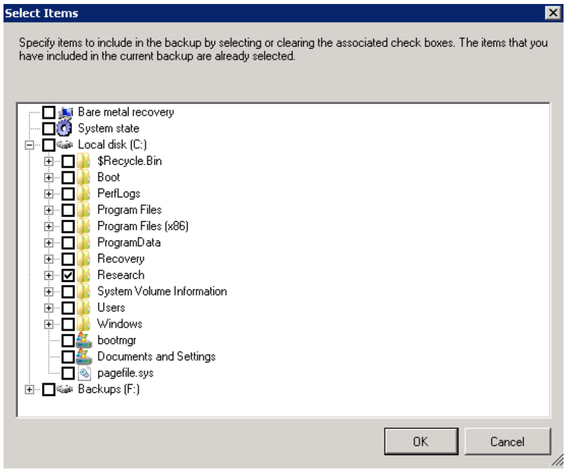


What type of configuration do you want to schedule?

( ) Full server (recommended)
I want to back up all my server data, applications and system state.
Backup size: 42.91 GB

(●) Custom
I want to choose custom volumes, files for backup.

5. On the Select Items for Backup screen, click the Add Files button.
6. On the resulting Select Items screen, select the folders that you want to back up, and click OK. In the following image below, the Research folder is selected.



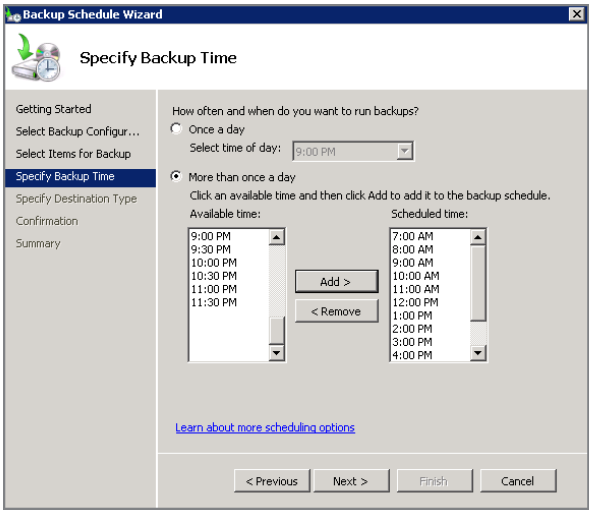
Specify items to include in the backup by selecting or clearing the associated check boxes. The items that you have included in the current backup are already selected.

- ☐ Bare metal recovery
- ☐ System state
- ☐ Local disk (C:)
  - ☐ $Recycle.Bin
  - ☐ Boot
  - ☐ PerfLogs
  - ☐ Program Files
  - ☐ Program Files (x86)
  - ☐ ProgramData
  - ☐ Recovery
  - ☑ Research
  - ☐ System Volume Information
  - ☐ Users
  - ☐ Windows
  - ☐ bootmgr
  - ☐ Documents and Settings
  - ☐ pagefile.sys
- ☐ Backups (F:)

7. Verify that the correct items are selected for backup on the Select Items for Backup screen, and click Next.

EXERCISE 11.1 *(continued)*

8. On the Specify Backup Time screen, select the More Than Once A Day radio button and then select the times of day that you want to perform the backups. The following screenshot shows hourly backups selected between 7 a.m. and 8 p.m.



How often and when do you want to run backups?

Once a day

Select time of day: 9:00 PM

[✓ More than once a day]

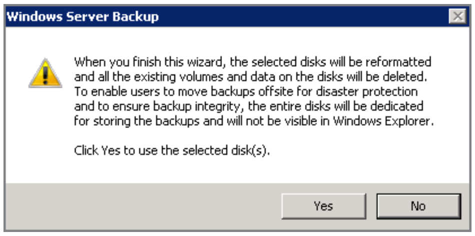
Click an available time and then click Add to add it to the backup schedule.

Available time: 9:00 PM, 9:30 PM, 10:00 PM, 10:30 PM, 11:00 PM, 11:30 PM

Scheduled time: 7:00 AM, 8:00 AM, 9:00 AM, 10:00 AM, 11:00 AM, 12:00 PM, 1:00 PM, 2:00 PM, 3:00 PM, 4:00 PM

9. On the Select Destination Type screen, choose the destination for your backup data to be stored and click Next.

10. On the Select Destination Disk screen, select the disk you want your backups to be copied to and click Next. This may result in a warning telling you that all of the selected disk will be formatted. This means all existing data on the disk will be deleted), so make sure that you select the correct disk as the backup target.



When you finish this wizard, the selected disks will be reformatted and all the existing volumes and data on the disks will be deleted. To enable users to move backups offsite for disaster protection and to ensure backup integrity, the entire disks will be dedicated for storing the backups and will not be visible in Windows Explorer.

Click Yes to use the selected disk(s).

11. On the confirmation screen, verify that all options are correct and click Finish.

This completes the steps required to schedule regular daily backups of a custom set of data on a Windows server when using the Windows Server Backup tool. It's worth noting that this tool provides only simple scheduling and doesn't allow you to specify different schedules for different days of the week such as weekends.

The Backup Window

The backup window is another item that relates to backup and business continuity. A *backup window* is a period of time—with a start and stop time—during which data can be backed up. For example, a backup window might include all noncore business hours for a certain business, such as 11 p.m. until 9 a.m. If that's the case, all backups have to start and complete within that window of time. However, many businesses have different backup windows for different applications, depending on core business hours for those applications.

A major reason for the backup window is that backups often negatively impact the performance of the system being backed up, so backing up a critical system during the core business day is usually not a good idea. It's important to get input from people close to the business when determining backup windows.

It's All about Recovery

Nobody cares about backups; people care only about restores. Backups are commonly considered the least sexy area of IT. Nobody thinks about backups when they're factoring things in for their applications, and nobody wants to dip their hands in their pockets to pay for backup infrastructure such as tapes and the like. However, everybody wants their data to be backed up, and everybody hits the roof if there aren't any backups when they need a restore operation! And believe me, if a business user needs data restored but there are no backups of that data, you will be in a world of hurt.

Backups are for the express purpose of being able to restore data after things such as user error/deletion, device failure, logical corruption, and so on. With that in mind, it's absolutely vital to be able to restore the data you're backing up. Backing up data, if you can't restore it, is pointless and will make you look incompetent in the eyes of your customers. As such, regular rehearsing of restoring data is vital. You should regularly test your capability to restore data that you have backed up.

Real World Scenario

The Importance of Testing Restore Operations

A company that I worked with was happily backing up its Microsoft Exchange environment until one day a restore of a crucial Exchange-related configuration file was requested. When an attempt was made to restore the file from the previous night's backup, the file was not found in the backup. This was assumed to be a missed file in that particular backup job, and an attempt was made to restore the file from the backup made the night before the previous night. Again the file was missing. This process was repeated, attempting to restore the file from week-end and month-end backup jobs; each time the file was missing. As a result, the file could not be restored, and the Exchange server had to be rebuilt.

It turned out that the Exchange server backups at this organization were being performed as standard filesystem backups and not with the Microsoft Exchange backup agents that work in conjunction with Microsoft Exchange to ensure application-consistent backups. This was immediately rectified, but it was too late to help with this specific restore request.

The takeaway point from this experience is that this company had never tested a restore of Exchange from their backups. When the time came that the company needed to rely on the backups, they were useless! So practice your restore operations!

NOTE

In addition to the importance of being able to recover backed-up data, there are some applications, such as databases, that need to be backed up frequently in order to be able to truncate log files. If you don't back these systems up regularly, the log files can fill up and force the database to stop.

Backup Architecture

The most common backup architectures are based on the tried and tested client-server architecture. Most backup architectures are composed of the following four components:

- Backup servers
- Backup clients
- Media servers
- Backup targets

The *backup server*, which sometimes goes by names such as *master server* or *cell manager*, is the brains of the backup environment. This server holds all the job definitions, the job schedule, the backup database and catalog, and all other metadata relating to the backup environment.

Backup clients are software agents that are installed on servers being backed up, and are responsible for getting the data off the client machine being backed up, and onto the storage medium. Application-specific backup agents such as Microsoft Exchange or Microsoft SQL Server also coordinate application consistency of backups, but more on that later.

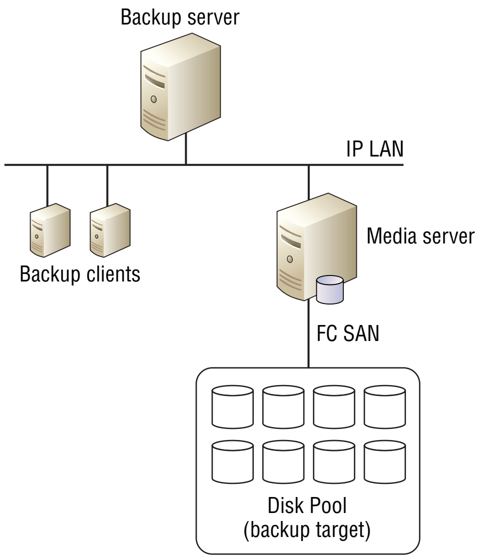
NOTE People in other departments of the company might not realize that the backup client agent is actually on the server, not the client. For most purposes, the distinction is negligible, but occasionally you might need to clarify this to your colleagues to avoid confusion.

Media servers connect to the backup target, and make it available to backup clients so that they can send data to the backup target. For example, a media server might be connected to a pool of storage over an FC network and make that storage available to backup clients over an IP LAN.

Backup targets range from tape drives and DVDs all the way up to large pools of disk-based storage or large tape libraries. Traditional approaches to backups primarily used tape as a backup target. More-modern approaches tend to favor disk-based mediums, including disk pools shared over LAN-based protocols such as SMB/CIFS and NFS, or even specialized disk arrays that imitate tape libraries, referred to as virtual tape libraries.

Figure 11.1 shows a typical backup environment with a single backup server, two backup clients, a single media server, and a disk pool.

FIGURE 11.1 Example backup environment



As you look at Figure 11.1, the high-level backup process would be as follows:

1. The backup server monitors the backup schedule.
2. When the schedule indicates that a particular client needs backing up, the backup server instructs the agent software on the backup client to execute a backup. As part of this instruction, the backup server tells the agent what data to back up and where to back it up to.
3. The backup server sends the data over the IP to the media server.
4. The media server channels the incoming backup data to the backup target over the FC SAN that it is connected to.

Backup Methods

The backup is a real industry old-timer. In fact, it's so old that it grew up with mainframe computers, and the two technologies are still close friends. Until recently, backup methods and technologies have resisted change. In fact, in many respects, backups still look the same as they did nearly 50 years ago. However, backup methods are finally succumbing to change, and there are an increasing number of methods for backing up servers and applications. This section presents some of the more popular methods and approaches.

Hot Backups

In the modern IT world, almost all backups are hot backups. *Hot backups*—sometimes called *online backups*— are taken while the application and server being backed up remain online and servicing users.

Hot backups are ideal for today's demands on IT resources, where servers and applications are expected to remain up and online at all times. Hot backups also help reduce administrative overhead and allow backup environments to scale because they don't require administrative stops and restarts of applications to be coordinated and managed just to cater to backups.

Most modern applications, such as Oracle Database and most of the Microsoft server products (Microsoft SQL Server, Microsoft Exchange Server, Microsoft SharePoint Server, and so on) all support hot backups. To provide this hot-backup feature, most applications place themselves into a hot-backup mode and utilize some form of point-in-time (PIT) technology that creates a frozen image of application data. This frozen image is then used for the backup. Quite often, snapshot technologies, including array-based snapshot technologies, are utilized as part of the PIT frozen image creation process.

Another matter of extreme importance in the world of hot backups is the ability to perform integrity checks against backups. Fortunately, most good applications that support hot backups have robust facilities to perform checksums, application-based integrity checks, and verifications of backup images. Doing this helps ensure that the backup is a good backup and will be useful as an image to restore from.

Offline Backups

Offline backups are the opposite of online hot backups, and hopefully they're a thing of the past. Offline backups require applications and databases to be taken offline for the duration of the backup. Obviously, this is far from ideal in the modern world, where businesses often operate 24/7 and cannot afford downtime of applications and services while performing regular backups.

LAN-Based Backups

LAN-based backups have been around for years and are perceived as cheap and convenient, but often low performance with a risk of impacting other traffic on the network.

LAN-based backups work by sending backup data over the LAN, and this LAN can be your main production network or a network dedicated to backup traffic. Dedicated networks can be virtual LANs (VLANs), where a dedicated L2 broadcast domain is created for the backup traffic, or it can be a physically dedicated network with its own network cards in hosts and network switches. This latter option obviously costs more money than a simple VLAN, but it offers the best and most predictable performance, as well as being the simplest to troubleshoot and the least likely to negatively impact other non-backup-related network traffic.

In a typical LAN-based backup configuration, the backup client will send data from the host being backed up, over the LAN, to the media server or directly to the storage media. The important point is that backup data is always sent over the LAN.

Figure 11.2 shows data being sent from a client over a dedicated backup LAN to a media server. The media server is then connected to the backup target via a Fibre Channel storage area network (FC SAN). In Figure 11.2, each server is connected to two LANs:

- Management LAN
- Backup LAN (dedicated to backup traffic)

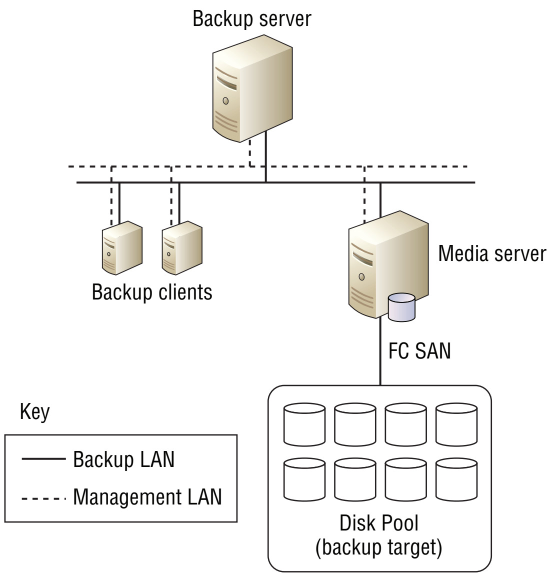
LAN-Free Backups (SAN Based)

A *LAN-free backup* architecture, as the name hopefully suggests, backs up data without crossing the LAN. Quite often, the data is passed from the backup client to the storage medium over the SAN.

NOTE

Sometimes LAN-free backups are referred to as *SAN backups* because they tend to back up data over the SAN rather than the LAN. They are also often referred to as *block-based backups* because they often don't understand, or care about, files. They simply back up all the blocks they're instructed to back up. This can be extremely fast when you are trying to back up millions of small files. However, it has an inherent downside. Block-based backups don't normally provide a smooth and simple way to restore individual files. If you want an individual file back, you may be required to restore the entire backup to a temporary area and then locate the required file and copy it back to its original location.

FIGURE 11.2 LAN-based backup



In comparison to LAN-based backups, LAN-free backups tend to offer higher performance and reliability, but at a higher cost. This is a real example of getting what you pay for. However, with the advent of 10G networking technologies, LAN-based backups can also be high performance, as well as high cost.

Being able to back data up at higher speeds can be important in environments where the backup window is a concern. If you are struggling to complete all of your backups within the allotted backup window, you may want to consider LAN-free backups or maybe even LAN-based backups over 10G Ethernet networks.

Figure 11.3 shows a LAN-free backup configuration in which the client being backed up has direct connectivity to a SAN-based tape library. In order to do this, the client being backed up usually must have either backup agent software that can talk directly to a tape library or an installation of the backup media server software.

In Figure 11.3, the backup server tells the agent software on the backup client to perform a backup to the tape library backup target. The backup data is then sent directly to the tape library over the SAN rather than the LAN. This lightens the load on the LAN and may enable organizations with existing investments in FC SAN technology to deploy a robust high-performance backup environment without having to deploy a dedicated backup LAN.

Serverless Backups

Serverless backups are a form of LAN-free backup, but they take the concept to the next step. They are not only LAN free, meaning they don't use the LAN to transport backup data, but also server free, meaning they don't utilize server resources either. Figure 11.4 shows this concept.

FIGURE 11.3 LAN-free backup

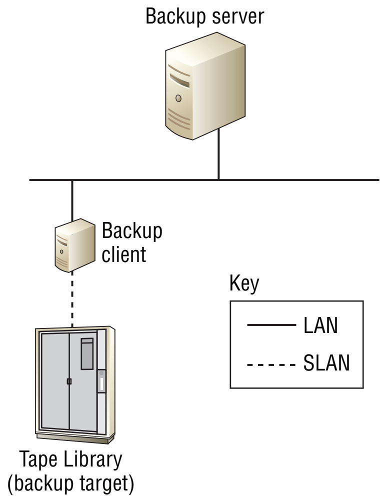
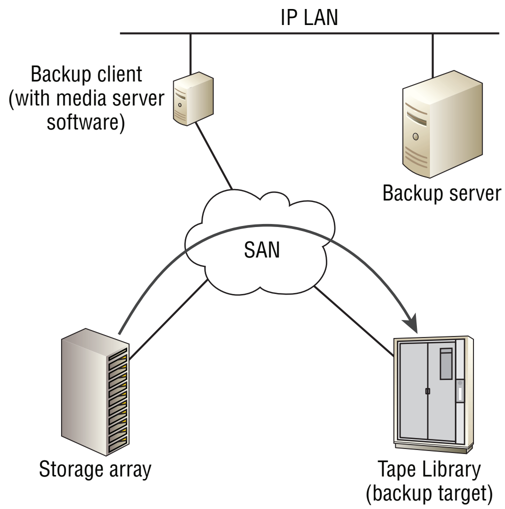


FIGURE 11.4 Serverless backup


IP LAN
Backup client (with media server software)
Backup server
SAN
Storage array
Tape Library (backup target)

Under the hood, serverless backups use the SCSI-based `EXTENDED COPY` command, which allows data to be copied directly from a source LUN to a destination LUN. And the important word here is *directly*. In serverless backup configurations, data is sent from the source LUN directly to the target LUN over the SAN, but without passing through the server that owns the data.

Because serverless backups work with SAN-based LUNs, and tape devices on a SAN are presented as LUNs, the target LUNs in a serverless backup configuration often are tape based.

NOTE The `EXTENDED COPY` command doesn't have to copy an entire LUN to another LUN. It copies specified ranges of blocks from the source LUN to the target LUN. Those ranges are the blocks containing the data that is being backed up.

Figure 11.4 shows a typical serverless backup configuration with the path that the backup data copy takes being highlighted by the thick line from the storage array through the SAN to the tape library. The process works roughly as follows:

1. The backup server tells the backup agent software on the backup client machine to initiate a serverless backup with the tape library specified as the target.
2. A SCSI `EXTENDED COPY` command is utilized to copy the data directly from the storage array to the tape library via the SAN. At no point does any backup data pass through the backup client.
3. Metadata about the job is passed over the IP LAN to the backup server. This metadata includes job details such as start time, completion time, completion code, and so on.

Although the data being backed up isn't sent over the LAN, the metadata concerning the backup is still sent over the LAN to the backup server and the backup client/agent. This is so that metadata such as job completion code, start time, end time, and so on can be stored on the backup server.

The advantages of serverless backups are the same as the advantages of LAN-free backups—high performance and high reliability, though at a potentially higher cost than LAN-based backups—with the added advantages of reducing the negative performance impact imposed on the server being backed up. After all, they're called *serverless* backups, so it's only to be expected that they don't touch or impact the *server*.

NDMP

Network Data Management Protocol (NDMP) is a protocol designed for standards-based efficient NAS backups. NDMP is similar to serverless backups in that backup data can be sent directly from the NAS device to the backup device without having to pass through a backup media server. Without NDMP, NAS-based file shares would have to be mounted on a backup media server to be able to be backed up. This configuration would drag data across the network (LAN) from the NAS to the backup media server and then possibly back over the network again to the backup device. With NDMP, there is no need to mount the NAS-based network share on the media server. Instead, data can be sent directly from the NAS device to the backup device, touching the network only once. This reduces the amount of load placed on the network.

Non-NAS-related network file servers, such as Windows and Linux servers, don't necessarily need NDMP because they can have backup agent software installed on them to enable them to back up directly to network-based backup devices. However, NAS arrays run custom operating systems that can't normally have backup agent software installed on them, and they therefore need a protocol such as NDMP.

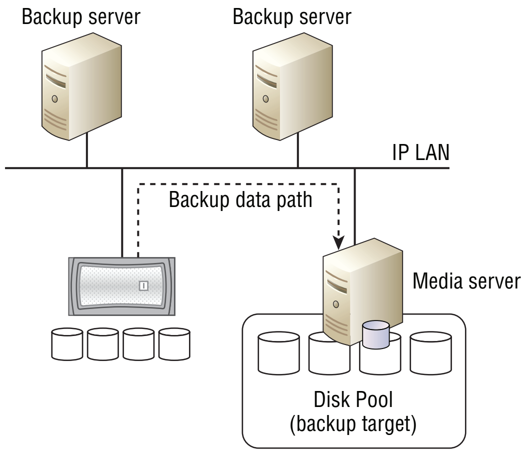
An important feature of NDMP backup configurations is direct-access restore (DAR). Direct-access restore enables the following:

- Selective file restore (including individual file restore)
- Faster recovery

However, there is no free lunch! A side effect of the faster and more granular restore capabilities provided by DAR is that backups take longer. This is because as each file is backed up, its location within the backup is recorded. This prevents you from having to read or restore the entire backup image to locate individual files or directories that you want to restore.

Figure 11.5 shows a simple NDMP configuration.

FIGURE 11.5 NDMP backup configuration



In Figure 11.5, the backup data is sent directly from the NAS device to the backup target disk pool over the LAN, as shown by the dotted line.

NOTE Backing up lots of small files can be a real killer—and not just for NDMP. For example, it will take a *lot* longer to back up 1,000,000 1 KB files than it will to back up a single 1,000,000 KB file (about 100 MB). A *lot* longer! One of the reasons is that each individual file has to be catalogued, and in some backup solutions, verified. It's similar to the difference between painting a single, large flat wall vs. painting the same surface area but on lots of smaller walls. With the large wall, you have only one set of corners and edges to paint, whereas painting lots of smaller walls massively increases the number of corners and edges you have to paint. If you have large filesystems with millions or billions of files and a small backup window that you are struggling to fit your backups into, then a block-based backup solution such as serverless backup might be a better option. However, block-based backups tend not to allow you to easily restore individual files.

Backup Types

In the modern world, there are many types of backups—full backups, incremental backups, differential backups, synthetic backups—all of which can work together to cater most business backup needs. Application awareness is another factor that needs to be considered when choosing a backup method. Let's take a look at them in more detail.

Full Backups

As the name suggests, full backups back up everything. They start at the top and don't finish until they get all the way to the bottom. The Storage Network Industry Association (SNIA) defines full backups as "a backup in which all of a defined set of data objects are copied, regardless of whether they have been modified since the last backup."

Like all things, full backups have their pros and cons.

On the positive side, they offer the best in terms of restore speed and restore simplicity. For example, if we back up an entire home drive with 250 GB of data on it every night and we want to recover the contents of our home drive from six days ago, we will probably need only one tape to perform the recovery. The process will require identifying the tape, loading it into the tape drive, and then restoring the data from the tape.

On the downside, full backups consume the most backup space on the backup target device, as well as the most network bandwidth and server resources. And they take the most time. Using the previous example of backing up our home drive every night, this means that every night we pull 250 GB of backup data across the network and store it to our backup device. Although that might not sound like a lot of data, it can quickly become a lot of data when you realize your organization has 100, 500, or over 1,000 home drives to back up each night. It also requires our entire home directory to be read as part of the backup process, placing additional read workload on the host and storage array hosting the home drive. So it's not all goodness when it comes to full backups.

Watch Out for What "Incremental" Means!

In common usage, and according to just about every commercial backup tool, an incremental backup is defined as a job that backs up files that have hanged since the last backup… and that "last backup" could be an incremental backup or a full backup. This means that a full recovery requires the most recent full backup plus all incremental backups since the full backup.

Sadly, at the time of writing, the SNIA defines incremental backups differently. According to the SNIA Storage Dictionary, a cumulative incremental backup (as opposed to a differential incremental backup) is a "backup in which all data objects modified since the last full backup are copied" (note the use of the word full). Because this book maps to the CompTIA Storage+ Powered by SNIA exam, the book uses the SNIA definition, especially in the next couple of sections. But don't be misled by that definition when you encounter "incremental backup" in non-SNIA, real-world contexts!

Cumulative Incremental Backups

A cumulative incremental backup-that's the SNIA term for what is commonly called a differential backup-is different from, but work in conjunction with, a full backup. The major difference between full and cumulative incremental backups is that cumulative incremental backups only back up data that has changed since the last full backup. So, in the first place, you need a full backup, and from there on, you can take cumulative incremental backups.

Most cumulative incremental backup solutions will back up an entire file even if only a single byte of the file has changed since the last backup. Let's look at an example based on our 250 GB home drive. Assume that we have a backup schedule that performs a full backup of our home drive every Friday evening, and then performs cumulative incremental backups every other day of the week. On Friday, we'll drag 250 GB over the network and store it on our backup device. Then on Saturday, we'll drag over the network and store to our backup device only files that have been updated since Friday night's full backup-probably not very much data at all. The same goes for every other backup prior to the next Friday night full backup; each cumulative incremental backup backs up only data that has changed since the previous Friday's full backup. And unless we change a lot of data in our home drive on a daily basis, this combination of full and cumulative incremental backups will save us a ton on backup storage space and network bandwidth.

That's great, but there's a downside too. Let's say it is Thursday, and we've deleted the entire contents of our home drive and need to restore it from the latest backup. The latest backup was Wednesday night, but Wednesday night's backup was only a cumulative incremental backup, so it contains only data that has changed since the full backup last Friday. So, in order to recover the entire contents of our home directory, we need the tapes from the Friday night full backup as well as the tape from Wednesday night's cumulative incremental backup. Our backup software then needs to coordinate the process so that what gets restored is an image of our home drive as it was Wednesday night.

It is slightly more complicated than taking a full backup every night, but when it comes to filesystem backups, most organizations decide that the benefits of using cumulative incremental backups during the week far outweigh the drawbacks.

Differential Incremental Backups

A differential incremental backup-again, that's the SNIA term-is a form of incremental backup. Like cumulative incremental backups, differential incremental backups work in conjunction with full backups. But instead of backing up all data that has changed since the last full backup, differential incremental backups back up only changed data since the last differential incremental. These are excellent at space efficiency, as each differential incremental backup will need to send even less data across the network to our backup device than a cumulative incremental backup. However, they have a downside. Differential incremental backups are the slowest and most complex when it comes to restore operations.

Let's take a look at how it works with our home drive example from the previous section. Let's say it is Thursday again, and we've deleted the entire contents of our home drive again, and need to restore it from the latest backup. The latest backup was Wednesday night. But this time Wednesday night's backup was a differential incremental backup, meaning it contains only data that has changed since the differential incremental backup the previous night (Tuesday night). And Tuesday night's backup was also a differential incremental backup, meaning it contains only data that has changed since Monday night's backup. And Monday night's backup was also a differential incremental backup. Hopefully, you get the picture. The net result is that in order to restore the entire contents of our home directory, we need the tapes from Friday night's full backup, as well all tapes from every subsequent differential incremental backup-Saturday, Sunday, Monday, Tuesday, and Wednesday! And again, the backup software then needs to coordinate the process so that what gets restored is an image of our home drive as it was Wednesday night.

This is way more complicated than even full and cumulative incremental backups. However, it is more space efficient and less of a burden on the network and host.

Synthetic Full Backups

A more recent form of backup—when compared to standard full, incremental, and differential backups—is the *synthetic full backup*. At a high level, a synthetic full backup works by taking a one-time full backup and then from that point on taking only incremental or differential backups. And for this reason, you may hear synthetic full backups referred to as a form of *incremental forever* backup system. But it's a little cleverer than this description might suggest. Instead of just taking incremental backups forever and always having to go back to the original full backup plus the latest differential incremental, periodically the synthetic full backup is created. And the synthetic full backup looks, feels, and works exactly like a normal full backup, only it isn't created by trawling every file in our home drive, hauling each one over the network again, and storing each one to our backup device again.

Let's look at our home drive example again. Let's assume we've decided to deploy a synthetic full approach to backing up our home directory. On the first Friday, we perform a traditional full backup. We read all 250 GB of data, pull it over the network, and store it as a single full backup *image* on our backup device. Great. Every subsequent night, we perform an incremental backup so that we pull only changed data over the network and store it to our backup device. Then, when the next Friday comes along, instead of performing another traditional full backup, with all of its associated overhead, we take last Friday's full backup image, take an up-to-date incremental, and then combine the previous full backup with the incremental backup we've just taken. And what we end up with is a backup image that is exactly what a traditional full backup image would look like, only we haven't had to read every file from our home drive and pull it over the network. And significantly, this synthetic full backup can then be used as the base for another week of incremental backups.

However, the way in which synthetic full backups are created can place additional stresses on the backup device, because the backup software has to read the entire contents of the last full backup from the backup device, while at the same time the backup device is writing out the new synthetic full backup image. This reading and writing can be a lot of hard work for a backup device.

Application-Aware Backups

Application-aware backups—sometimes called *application-consistent backups*—are a form of hot backup and are vital if you want to be able to restore your applications from your backups.

It's one thing to back up standard filesystem data, such as home drives and file shares, as flat files. But if you try to back up databases or applications such as Microsoft Exchange as flat files without coordinating the backup with the application, you almost certainly won't be able to recover your application from your backups!

Application-aware backups usually require installing special agent software on the application server, such as installing an Exchange Server agent on the Exchange server. This agent does the usual job of communicating with the backup server and sending data to the backup device, but it also provides the intelligence required to back up the application in a consistent state.

Real World Scenario

Make Sure You Test before You Make Changes

One company had experienced significant growth and was looking for options to reduce the network bandwidth consumed by backup traffic. Because they had started out as a small company, their LAN-based backup design didn't have a dedicated backup LAN. They looked to implement synthetic full backups.

However, their backup target devices were disk-based deduplication appliances. During the testing phase, before rolling out synthetic full backups, they noticed that backup performance was significantly impacted in a negative way when creating the synthetic full images. Upon closer inspection, it was realized that the way in which synthetic full backup images were created was generating a workload pattern that the deduplication backup appliances couldn't deal with; intensive reading and writing at the same time is bad enough for a disk-based storage system, but is compounded when deduplication is thrown into the mix. This is because when data is read back from a deduplication appliance, it has to be rehydrated (un-deduplicated), adding significantly to the disk thrashing that was already taking place. The end result in this case was reverting back to traditional full backups and looking at implementing a backup LAN.

Many applications provide a mechanism to back themselves up both hot and in a consistent state. In the Microsoft world, the Volume Shadow Copy Service (VSS) is a technology that allows Microsoft applications to be backed up in that hot and consistent state. Similar options exist for some non-Microsoft options, such as RMAN for Oracle. However, as VSS is so popular and well-known, let's take a quick look at how it works.

NOTE It is technically possible for any Windows application to plug into and leverage the VSS framework for application-aware backups. For example, Oracle running on Windows can use VSS for application-aware backups.

Microsoft Volume Shadow Copy Service

In the Microsoft VSS space, the backup software—which is known as a *requestor* in VSS parlance—instructs the VSS service running on a server that it wants to back up an application on that server. For example, if the intent is to back up Microsoft SQL Server, the backup software (VSS requestor) will tell VSS on the machine that is running SQL Server that it wants to back up SQL Server. VSS then tells the application (SQL Server) to prepare itself for a hot backup. The application does this by completing in-flight I/O, flushing local buffers, applying logs, and setting any application flags required for smooth recovery. The application then reports its status back to VSS. VSS then instructs the application to quiesce (freeze application I/O) while the snapshot is created. Once the snapshot is taken, VSS instructs the application (SQL Server) to resume operations. All application I/O that was frozen during the quiesce is completed, and normal operations resume. The backup is then taken based on the snapshot created by VSS. But it's important to understand that even during the quiesce period, while the application remains up and running and serving user requests, the application I/O that is frozen is still occurring but not being committed to disk until the quiesce period ends. The quiesce period usually lasts for only a few seconds while the snapshot is taken.

The end result is an application-consistent backup—one that is guaranteed to be viable as a recovery option—that was taken while the application was online and working. No users or clients of the application will be aware that the application was just placed in hot backup mode and a backup was taken.

Backup Targets and Devices

All backups need to be stored somewhere. Technically speaking, we refer to the destination of backups as *backup targets,* though sometimes we use the term *backup device*.

In the past, backup targets were almost always tape based, using technologies such as tape drives and tape libraries. Nowadays, though, there are all kinds of options, including, but not limited to, the following:

- Tape devices
- Virtual tape libraries
- Disk pools
- Cloud services

Let's take a closer look at each.

Backup-to-Tape

Backup-to-tape—sometimes referred to by the SNIA as *B2T*—is probably the oldest and most common form of backup.

Advantages of Tape

Backup-to-tape consists of backing data up to linear tape devices. Although tape might seem very much like an ancient technology, modern tape drives and tape libraries have many positive features, including these:

- High capacity
- High sequential performance
- Low power

On the performance front, the nature of tape is almost perfectly suited to back up performance requirements. The typical workload of a backup job is streaming sequential data, and tapes thrive on sequential workloads. However, tapes suck at random performance. But random performance isn't the top requirement in most backup solutions.

Disadvantages of Tape

There are also some drawbacks to tape as a backup medium:

- Media degradation
- Technology refresh
- Awkward for some restore types

As a form of magnetic storage medium, tapes are subject to degradation over time, especially if they aren't stored in optimal conditions—heat controlled, humidity controlled, and so on. Storing tapes in such ideal conditions costs money. Secure off-site storage of tapes, sometimes referred to as *vaulting*, can be even more costly.

There's also the concern that when you come to restore data from tapes in 10 years' time, even though you've stored the tapes in optimal conditions, you no longer have any tape drives or software that can read them.

In addition, tapes are great at restoring data from full backups, but if you have to perform restores based on differential backups, they can be cumbersome. For example, loading and unloading multiple tapes is awkward, as is fast-forwarding and rewinding back and forth through the tape reel in order to locate the right place on the reel where the data you want is stored.

Because of these concerns, as well as the drop in the cost of spinning disk media, many people are starting to move away from tape for backups. That said, it is still popular, and there is still a market for tape.

LTO Technology

All kinds of tape technologies are available, each requiring a specific type of tape drive and sometimes library. However, the most popular tape format in the open-systems space is the Linear Tape-Open (LTO) format.

Within the LTO standards, there are various versions of LTO tapes, each designated by a number. Current LTO versions include LTO-1, LTO-2, LTO-3, LTO-4, LTO-5, and LTO-6.

According to the LTO standards, they have the performance, capacity specifications, and other features shown in Table 11.1.

TABLE 11.1 LTO specifications

Features	LTO-1	LTO-2	LTO-3	LTO-4	LTO-5	LTO-6
Uncompressed capacity	100 GB	200 GB	400 GB	800 GB	1.5 TB	2.5 TB
Compressed capacity	200 GB	400 GB	800 GB	1.6 TB	3 TB	6.25 TB
Uncompressed speed (MBps)	20	40	80	120	140	160
Compressed speed (MBps)	40	80	160	240	280	400
Compression ratio	2:1	2:1	2:1	2:1	2:1	2.5:1
Encryption	No	No	No	Yes	Yes	Yes
WORM	No	No	Yes	Yes	Yes	Yes

The LTO standards also mandate the following backward compatibility between drives and media:

- An LTO drive will be able to read data from tapes in its own generation, plus the two previous generations.
- An LTO drive will be able to write data to tapes in its own generation, plus the generation immediately preceding it.

Shoe-Shining

Shoe-shining is a phenomenon in the tape backup world that occurs when the speed at which data being sent to the tape drive doesn't match the speed that the tape drive is running at. When this occurs, the tape drive has to occasionally either slow down or momentarily stop while the data coming in from the media server builds up again. This starting and stopping of the tape reel, or slowing down and then speeding up of the tape, looks a bit like the back-and-forth action of shining shoes with a cloth. While occasional shoe-shining will slow down backup jobs, excessive shoe-shining can cause early wear and tear on the tape cartridge and the heads and motor of the tape drive.

Multiplexing

Multiplexing sends data from multiple backup jobs to a single tape drive. The objective is to increase the speed at which data is sent to a tape drive, allowing the tape drive to operate at maximum speed. This is great at helping to avoid shoe-shining.

However, multiplexing can affect restore performance. This is because the multiple streams of data that were multiplexed to the same tape drive are all interleaved next to each other. When you come to restore the data from one of the jobs (one of the streams), all of the data from all of the multiplexed jobs has to be read, and the data from the jobs that you're not restoring from has to be discarded, reducing the overall performance of the drive and the restore operation.

As a general rule of thumb, multiplexing enables faster backups but slower restores. When considering deploying multiplexing, you need to weigh the drawback of slower restores against the benefit of performing multiple backup jobs.

Tape Compression

Most tape drive technologies, including LTO, support native compression. The LTO specification uses a form of compression known as Streaming Lossless Data Compression (SLDC). Although SLDC is not the most efficient compression algorithm available, it is fast and recognized as a good balance between good compression and good performance.

NOTE

Encrypted data doesn't compress well. Attempting to compress data that is encrypted can result in the data set being larger than it was before the compression attempt was made. Fortunately, most backup technologies that encrypt data before sending it to tape mark it as encrypted so that no attempt is made to compress it.

As you can see in Table 11.1, each version of the LTO standard lists its native compression ratio as a feature. This native tape compression is a feature that is always on.

Obviously, compression can significantly help reduce the number of tapes that you need to support your backup estate, helping bring costs down.

Tape Encryption

Over the last few years, there have been several significant losses of backup tapes that have been high profile enough to make the trade press and sometimes the national press. This has served to highlight the need to encrypt *all* data that will leave the confined walls of our corporate data centers. And tape backups are no exception. Make sure you encrypt your tapes!

The LTO standards define encryption-capable tape drives from LTO-4 and onward, but they don't mandate that all LTO-4 drives support encryption. So make sure you check the specifications of any new LTO tape drives you purchase if you're planning to use tape drive–based encryption.

The encryption algorithm used by LTO is a symmetric block cipher (256-bit AES-GCM) that uses the same key to encrypt and decrypt the data. And with LTO drive based–encryption, a third-party key manager will be required for key management. Fortunately, most good backup applications will perform the task of key management.

On the performance front, most good drives will be able to perform encryption at full line-speed within the drive.

Because encrypted data can't be compressed, encryption should be performed after data is compressed, which is exactly what happens with LTO drive-based encryption.

Virtual Tape Library

The rise of the *virtual tape library (VTL)* was a major step forward for the backup industry. The VTL was the first major step away from tape-based backup targets to disk backup targets, and it has paved the way for more-native approaches to backing data up to disk-based targets.

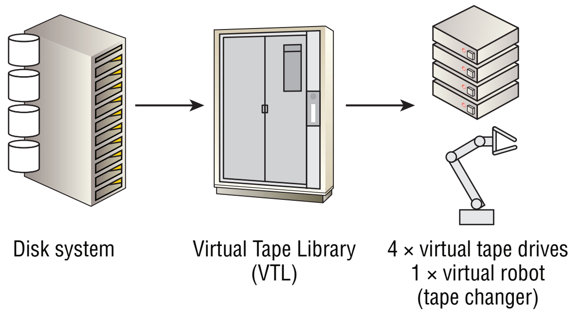
At the time that VTL emerged, a lot of companies already had established practices, procedures, and policies around working with tape as the backup medium. In these cases, VTL technologies helped enable these companies to start using disk-based backup targets without having to change all of their existing processes and practices.

Today, nearly every backup application on the planet supports native backup to disk without the need for tape emulation (VTL). In fact, most modern backup applications have more-robust backup-to-disk options than they do backup-to-tape, with some backup apps not even supporting backup-to-tape!

At a high level, VTL technologies take disk-based storage arrays and carve them up to appear as tape libraries and tape drives—hence the name virtual tape library. The initial requirement for this virtualization is to enable existing backup software to back up to a disk-based backup target without having to be modified to understand disk pools—basically taking a disk-based backup target and making it look and feel exactly like a tape library so that existing backup software could talk to it.

Figure 11.6 shows a disk array being virtualized into a VTL configured with four virtual tape drives and a single robot.

FIGURE 11.6 VTL emulation



NOTE: Strictly speaking, VTL technology is not a true backup-to-disk (B2D) technology. The major reason is that it takes a disk-based system and makes it look like an old-fashioned tape library. It is disk pretending to be tape, so doesn't strictly qualify as backup-to-disk.

One important technology that VTL solutions pioneered and pushed market acceptance of is data deduplication. Although not a feature restricted only to VTL technology, VTL solutions almost always perform native deduplication.

There are two major reason for this. First, backup workloads are ideally suited to deduplication. We back up the same data sets again and again, and often much of the data hasn't changed and so will deduplicate very well. Second, in order to be price competitive with traditional tape libraries, disk-based VTL systems needed a technology that allowed them to store more data on less tin.

NOTE For detailed coverage of deduplication technologies, see Chapter 10, "Capacity Optimization Technologies."

Backup-to-Disk

Backup-to-disk (B2D)—sometimes called *disk-to-disk (D2D)*—is similar to backup-to-tape (B2T), but the target medium is disk rather than tape. Disk has a few advantages over tape:

- Superior random access
- Superior concurrent read and write operation
- Better reliability

Although random access isn't always seen as a crucial performance consideration with reference to backup (after all, a lot of backup workloads are heavily sequential), there can still be benefits to having good random performance. One such advantage is small file restores (which are one of the most common types of restore operation). With disk-based restores, the system can almost immediately locate the data required and begin the restore. With tape-based systems, it takes time to load the tape and then to fast-forward or rewind to the desired location.

Disk-based systems are also far superior at concurrent read and write operations. This can be helpful if creating synthetic full backups, where the system is both reading and writing during the creation of the synthetic full backup image (reading the last full backup and required incrementals, while at the same time writing out the synthetic full image).

Another place where it can help is concurrent backup (write) and restore (read) operations, though to be fair, it's uncommon to be both backing up and restoring from the same tape at the same time. However, if you don't have a lot of tape drives, you may well find yourself needing to unload a tape that you were backing up to in order to load a tape you need to restore from. Disk-based backup targets, with their superior concurrent read and write performance, will help you here.

Generally speaking, backup to disk can often offer faster restore times. For example, there is no waiting around for tapes to be loaded and spooled into position. Also, the native sequential performance of some tape drives can be hamstrung by multiplexed backups, meaning that tape-based restores aren't always as fast as they could be.

Disk-based backup targets offer a more reliable backup medium than tape does. This is based on the fact that, although disks and tapes are both susceptible to failure, disk arrays are almost always protected by RAID technologies. The net result is that when disks fail in a disk-based system, no data is lost. Tapes, on the other hand, are rarely, if ever, protected because backup jobs are rarely duplicated to multiple tapes to cater to failed tape scenarios.

An important consideration when performing backup to disk is that backup software needs to be able to understand disk-based backup targets such as disk pools, as they are significantly different from tape drives and tape cartridges. For one thing, there is no concept of tape cartridges, scratch pools, bar codes, and other tape-related aspects. However, all good modern backup applications have extensive support for disk-based backup targets.

Backup to disk is usually done over IP networks and provides a filesystem interface, such as NFS and SMB/CIFS, to the disk pool. It also allows multiple backup and restore operations to simultaneously access the disk pool.

In most environments, using backup-to-disk does not signal the end of tape. In actual fact, backup-to-disk is being used to augment existing tape-based backup solutions, improving them significantly. A common approach is to stage backups to disk and then eventually copy them off to tape, in a scheme often referred to as disk-to-disk-to-tape (D2D2T). Backups are made to disk targets and kept there up to a week or so. During this time, they are most likely to be required for restore operations, and because they're on disk, they can give excellent restore times. After this period, jobs are moved off to tape, where they are then taken off-site and stored for long-term retention. One downside to this approach is the need for an additional copy job to copy the backup images from the disk target to the tape target.

Backup to the Cloud

With all things in IT these days, the cloud is making inroads. The backup space is no exception.

With the cloud, you generally use services and pay based on how much you use the service. This commercial model is often referred to as a *consumption model* or *consumption-based billing*. You can also quickly and easily flex up or down the amount of a service you consume using the cloud.

For backup and recovery, the cloud is interesting as a backup target. In one sense it's almost ideal, as you don't need fast or frequent access to your backups after they're stored to the cloud. However, the big question is how you get your backed-up data to the cloud. A common approach is to deploy a disk-to-disk-to-cloud (D2D2C) architecture. This is similar to disk-to-disk-to-tape (D2D2T), with the tape being replaced by the cloud. In this architecture, the disk-to-disk portion still occurs within your data center, utilizes deduplication technology, and transfers only deduplicated data to the cloud for long-term retention, where it's not expected to be required for restore operations as often as the data that is on premise on the intermediary disk backup platform. Deduplication technology and synthetic full backups, or other incremental forever approaches, also help here. It should also be noted in this D2D2C architecture that only a portion of the backup estate exists in the cloud.

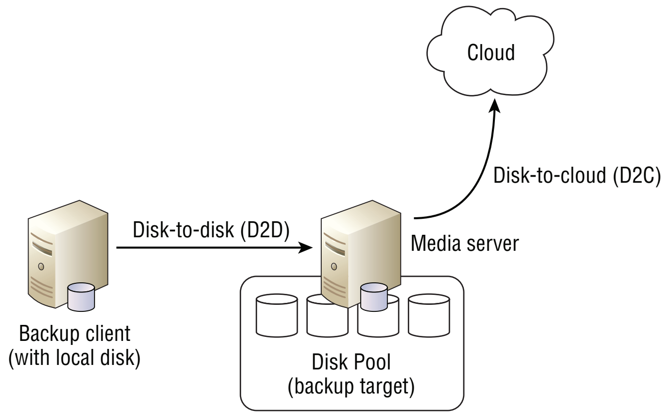
D2D2C architecture is shown in Figure 11.7. The process is broken into two steps:

1. Backing up data from disk to disk
2. Copying the same data from disk to the cloud

The two steps are combined for the complete D2D2C model. There will usually be a few days between step 1 and step 2, during which time the backup image is kept on site and on disk for fast restore. After a small period of time, such as a week or so, the backup images can be moved out to the cloud.

However, even with local disk-based backup targets and deduplication, if the amount of data you back up each month is too much, utilizing the cloud as a backup target just isn't going to work for you. It will simply take too long for you to get your data up to the cloud. In addition, you have to consider how long it will take to get it back if you need to perform a large-scale restore. Obviously, it depends on your deduplication ratios and network connection to your cloud backup provider (Internet links usually won't cut it), but if you're backing up about 10 TB or more each month, using the cloud as a backup target might prove challenging.

FIGURE 11.7 Disk-to-disk-to-cloud



Even with all of these considerations, Gartner expects that 40 percent of the $30 billion + backup market will move to the cloud by 2016. In late 2013, Amazon Web Services (the largest public cloud provider on the planet) introduced a VTL gateway to its cloud storage platforms, hugely simplifying the road to incorporating cloud storage into corporate backup environments.

If you decide to go with the cloud as a backup target, you need to make sure you encrypt all data that you send to the cloud and that you own the encryption keys and keep them outside that cloud service. You will also want a clearly defined exit plan that allows you to get your data back in the event that you are not happy with the service.

Backup Retention Policies

Backup environments require *backup retention policies*. These policies dictate how long backup images are kept. For example, you may want to keep a weekly full backup image for 365 days, but keep daily incremental images for only 28 days.

The most popular retention scheme in use is the Grandfather-Father-Son (GFS) scheme. The GFS scheme is based on three generations of backups, as follows:

Grandfather: Monthly backups

Father: Weekly backups

Son: Daily backups

GFS schemes also apply retention policies against each generation of backup. Probably to best understand how this works, you should walk through the steps of creating such a policy. Exercise 11.2 shows how you can set up a basic backup retention policy.

EXERCISE 11.2

Creating a Backup Retention Policy

In this exercise, you will set up a basic backup retention policy. You'll use the popular GFS scheme.

1. Determine the three generations of backups that you will use. Go ahead and use this scheme, which is fairly practical for many circumstances:
   - Grandfather: Monthly backups
   - Father: Weekly backups
   - Son: Daily backups
2. Determine your GFS retention policy to apply to each generation of backup. Make these your retention policies for your backups:
   - Monthly backups: Keep for 60 months (5 years)
   - Weekly backups: Keep for 52 weeks (1 year)
   - Daily backups: Keep for 28 days
3. Design a backup policy based on this particular retention scheme. Here is a basic approach you can use:
   - **a.** Take full backups every Friday night and class each one as a weekly backup.
   - **b.** Designate the first weekly backup of each calendar month as a monthly backup. Each of these will be expired based on the preceding policy (expire weekly backups after 52 weeks and monthly backups after 60 months).
   - **c.** On Saturday through Thursday, incremental backups are taken and expired after 28 days.
   - **d.** If you're using tape as your backup media, you can add into the mix the requirement to store weekly and monthly full backups off-site, but keep daily incremental backups on site.

This is the basis of a simple but powerful backup retention policy that we could publish to our users as part of a service-level agreement as follows:

1. All data from the last 28 days can be restored from the previous night's backup (RPO), and can be restored within one working day (RTO).
2. All data between 28 and 365 days old can be restored to the nearest Friday night backup of the data (RPO), and can be restored within two working days (RTO).
3. All data between 1 year and 5 years old can be restored to the nearest first Friday of the month (RPO), and can be restored within two working days (RTO).

In this example SLA, we highlight references to RPOs and RTOs. For example, in the first item we state that the point in time that we can recover data from the last 28 days is the previous night, giving us an RPO of no greater than 24 hours. We also state it will take us no longer than one business day to get this data back, giving us an RTO of one business day.

Obviously, this backup retention policy won't fit all scenarios, but it is a good starting point that is easy to adjust.

When relying on incremental and differential backups, make sure that you keep the full backups that these rely on (the last full backup before the incremental or differential). In a GFS-based system like the one described in Exercise 11.2, this will always be the case, as we keep weekly fully backups for one year and monthly full backups for five years.

NOTE Defining SLAs should be done in consultation with the business or IT personnel designated as the liaison with the business. The agreements should also be published to both IT and the business and should be regularly reviewed.

Archiving

First, and foremost, archiving and backups are not the same! It's like saying cars and airplanes are the same, because they both transport people from one place to another. But try taking a plane to the grocery store, or using your car to take a trip to somewhere on the other side of the world. You *could* do both, but neither are ideal. The same goes for backups and archiving technologies and their use cases.

Backups and Archiving Comparison

Let's quickly compare and contrast backups and archiving.

For the most part¸ *backups* are intended to protect data that is *currently* or *recently in use*. Backups provide medium to fast recovery of this data, usually in the event that it's lost, deleted, or corrupted. It is kind of like an insurance policy in case you lose your data and need it back quickly in order to keep running the business.

Backups deal with *operational* recovery of files, folders, applications, and services. The typical use cases include the following:

- User-requested restores
- Disaster recovery (DR) and business continuity (BC)

On the topic of user-requested restores of backups, we've probably all been there. A user calls up or logs a ticket with the help desk, asking for a specific file or folder to be restored, usually because they've just deleted it. Previous backups of the deleted file or folder are then used to restore the lost data. We all cross our fingers, hoping that there's a valid backup to restore from.

Using backups for DR is different. As a last resort—after attempting potentially quicker methods of DR such as replication—backups can be used to recover entire systems, either at the local or a remote site.

Archiving is used to efficiently store data that is *no longer used*, or only very infrequently used, but needs keeping. In the case of email archiving, copies of emails are made as they are sent and received, so they can't be tampered with before entering the archive. The most common reason for archiving and retrieving data is legal compliance requirements. Most data stored in an archive needs to be tamper proof so it can be guaranteed to be unchanged from the time it was stored.

When comparing backups with archives, it's helpful to think of backups as being used to put lost data back to its original location so it can be used for its original purpose. Objects that are backed up are indexed according to their location. Archives, on the other hand, are used to locate data based on its content, and usually to copy the data to a new location where it can be used for a different purpose—often corporate or legal compliance. Archives are usually full-text indexed so you can locate items based on their content rather than their location.

NOTE Archive systems that provide tamper-proof storage of data are said to provide *content assurance* or *content authenticity*. The term *nonrepudiation* is also used in reference to archive systems that provide content authenticity, indicating that you cannot deny the authenticity of the data.

Even though backups and archives are not the same, they do complement each other. One example is that archived data generally doesn't need to be backed up like nonarchived data does. This is because data in the archive doesn't change. As long as it's backed up and then stored on a reliable archive platform, it may never need backing up again. This means that moving old, unused data to an archiving solution can help trim back the amount of data that you're backing up each month.

Compliance

Regulatory compliance is no doubt the single-largest driving factor behind the uptake of archiving solutions. Most developed countries have legal requirements around the storage, retrieval, and eventual deletion of data, especially data of a personal or business-sensitive nature. In the United States, the Sarbanes-Oxley Act of 2002 (SOX) is one of the more well-known examples. Others include Payment Card Industry Data Security Standard (PCI DSS), and various data protection laws that are legal in different countries and territories across the world. If your business is a global business, you will likely need to incorporate and comply with multiple compliance regulations.

Across the world, though, most legal requirements call for data in an archive to meet the following standards:

- Be tamper proof
- Be deleted after a given time
- Provide an audit trail

In the storage world, various technologies can provide tamper-proof capability, and we often refer to them as write once, read many (WORM) technologies. Some, such as some DVD and Blu-ray Disc technologies, provide this feature natively. Others, such as disk-based solutions, need to either be designed in a specific way or have special firmware written for them that make the data they store tamper proof. One such technology in the ever popular disk-based compliance archiving space is content-addressable storage (CAS).

NOTE For an in-depth discussion of content addressable storage, see Chapter 7, "Files, NAS, and Objects."

Archiving solutions are also often required to provide detailed audit logs showing who has access to data in the archive and who *has* accessed it.

Also, with some regulations, it's just as important to delete sensitive data after a certain amount of time as it is to keep it in the first place. Many regulations state that after a specified amount of time, data *must* be purged from the archive.

How long your company keeps copies of data will be dictated by legal and compliance people at the company. However, not keeping data for long enough can have negative side effects. For example, if you have a policy stating that backups of email will be kept for only 30 days, you obviously cannot restore emails from more than 30 days ago.

NOTE In the real world, an increasing number of companies are chomping at the bit to delete and purge data as soon as possible. There are two main reasons driving this behavior. First, deleting data frees up valuable space in the archive. Second, data kept in archives, for regulatory reasons, can be and often is used against the company in litigation and other situations that can impose fines and similar penalties on the company. For these two reasons, many companies are keen to get rid of data as soon as possible.

Discovery

Archives need to be indexed and quickly searchable. When a request is made to restore lost or deleted files from a previous night's backup, the user logging the call knows the filename and file location. In contrast, when a file or email is subpoenaed, the information provided will most likely not be as specific. For example, if a court subpoenas an email, it will most likely ask for something along the lines of “an email sent by XYZ executive between July and August 2010 regarding discounted pricing of XYZ product.” This is far more vague than a typical user requesting a restore of a file. Because of this, it’s a top requirement of an archiving solution to be quickly and extensively searchable. If you can’t locate and retrieve data that has been requested by a court within the time the regulations specify, your business will almost certainly be landed with a fine.

Archive Media

Disk is ideally suited as an archive medium. It provides large capacity, good random access, and reliability when deployed with RAID technologies. And it’s getting cheaper all the time.

Because data that is stored in an archive is stored only once, it’s vital that the medium that the archived data is stored on is reliable. If that media becomes corrupted or fails, you’ll lose your only copy of that data in the archive, leaving you in a very tight place, especially if you have to respond to a legal request for that data. On the reliability front, disk-based archives are almost always protected by RAID-type technologies, making them extremely reliable, and significantly, more reliable than tape.

Disk-based archives are also a lot better than tape when it comes to technology refreshes. Although both tape and disk need refreshing every few years, tape technology refresh is particularly painful and labor-intensive. Nobody wants to do it. Consequently, tape refresh often gets overlooked until you find yourself with very old backups on very old tapes that you can no longer recover from. And as tapes usually get stored off-site, the truism *out of sight, out of mind* applies, and we often forget that we need to refresh them.

On the other hand, disk-based archives sit in the middle of your data center and suck power every single day, meaning you’re less likely to forget about them. Also, the process of migrating data from a disk-based archive to a newer disk-based archive is far simpler and far less work than copying old tapes to new tapes.

On the topic of power consumption and associated operational expenditure costs, some disk-based archives support spin-down and massive array of idle disks (MAID) technologies, where unused disk drives are spun down (turned off or put into low-power mode) to reduce overall power consumption of the archive. When they are needed, they are spun back up. However, this extra time taken to spin them back up is negligible compared to the time required to recall tapes from off-site storage.

Optical media, such as DVD and Blu-ray, are options as archive media, although they are rarely seen in the real world. They’re both removable, meaning they can be vaulted, and they both offer native WORM capability. They can also be cheap. However, archive solutions are very long-term investments, and you need to think extremely hard about how your archive solution will look in 10 years. This is one area in the real world where optical media is falling short. These are all valid questions that you need to seriously consider:

- How long will your chosen optical media be around for?
- How reliable will it be in 10 years?
- How long will the payers be around for?

Optical media also struggled with capacity and performance. Tapes and disks can store far more data and often provide faster access, and with disks, you don't waste time having to mount and unmount the media.

Real World Scenario

The Dificulty of Using Backups as an Archive

A U.S.-based retail company was subpoenaed for some data relating to an investigation, but they didn't have a proper archiving solution in place. Instead, they had to rely on an old backup to recover the data. Because the data being requested was from several years ago, the backup tapes that the data was thought to be on were stored off-site with a third party and had to be recalled to site. There was a delay of an extra day in recalling the tapes because they were from so long ago. Then when the tapes arrived back at the site, they used an old format of tape (SDLT), and the company no longer had any tape drives at the data center that could read that format. Fortunately, a suitable tape drive was located, based in a branch office, and arrangements were made for it to be shipped to the data center ASAP—meaning local backups at the branch office had to be disabled until the SDLT drive was returned. When the drive arrived at the data center, a new media server had to be built to house the tape drive. Also, because the data was so old, each individual tape had to be read in order to index its contents. As several tapes had been recalled, this took another two to three days. In the end, the data was recovered and provided to the legal department.

On top of all of this overhead, there was an increased risk that the tapes would be unreadable because of media degradation. In this particular instance, this didn't happen, but there was a good chance that after all the efforts of the IT staff in recalling tapes, borrowing the tape drive, and building new servers, the tapes would have been unreadable.

Restoring the data also would have been made even more complicated had the company changed backup applications since the data had been backed up!

This kind of experience is in stark contrast to using a purpose-built archiving solution, where the data sits on reliable media that is maintained (disks are scrubbed and RAID protected), indexed, and searchable, and data is accessed over industry-standard protocols such as SMB/CIFS and NFS.

Summary

In this chapter we set the scene by discussing why we back up and why backups are so important, and defined some important terms such as recovery point objective, recovery time objective, and backup window. We then stressed the importance of being able to restore from backups. After that, we talked in detail about the major types of backup, including hot backups, LAN-based backups, LAN-free backups, serverless backups, and NDMP backups. We then discussed the differences between full backups, incremental backups, and differential backups, and how they can be effectively used as part of a backup retention scheme. We then moved on to discussing different backup target technologies such as tape, disk, and the cloud, and finished the chapter talking about archiving technologies and compliance archives.

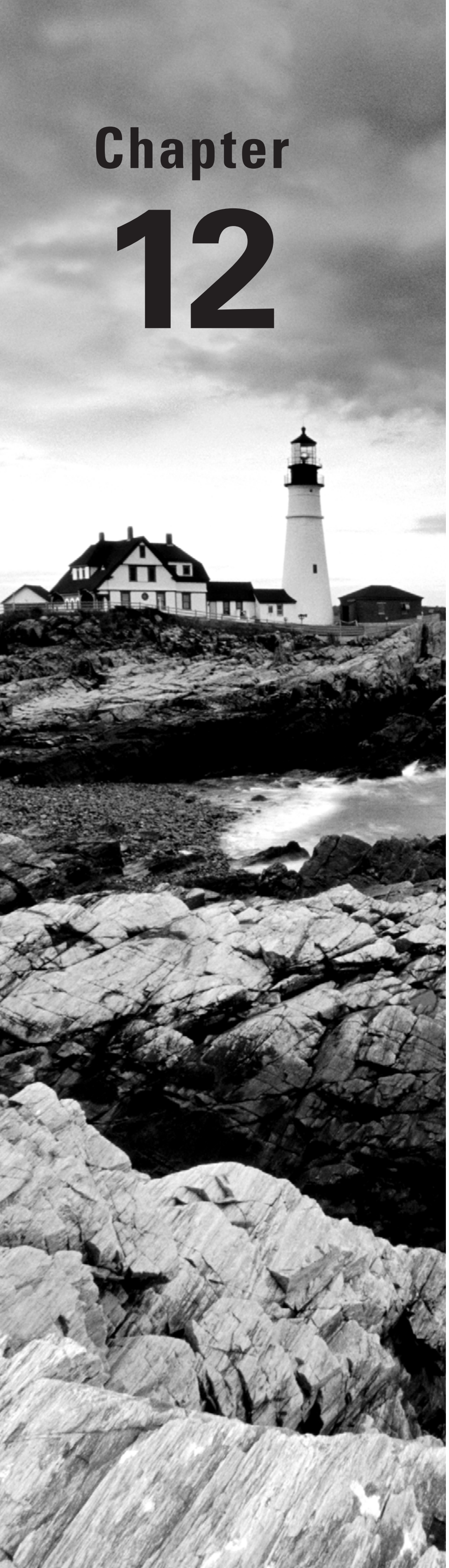
Chapter Essentials

Backups and Business Continuity Backups are an important part of disaster recovery and business continuity plans. It's absolutely vital that backups are reliable and can be easily and reliably recovered from. Take the time to test your restore procedures so that you don't end up trying to recover data from a particular type of backup in the heat of an emergency.

Hot Backups Hot backups are application-consistent backups taken while a system remains online. No system downtime is required for backups, and the backup is guaranteed by the application to be consistent and therefore useful as a recovery point. Most popular applications and databases support a hot backup mode.

Synthetic Full Backups Synthetic full backups reduce the client and network load required to create full backup images. Rather than trawl the entire contents of the client being backed up and drag all the associated data across the network to the backup target, synthetic full backups work by taking a previous full backup image, already stored on the backup target, and joining it with a new incremental image. The resulting joined image is exactly what you would have achieved had you taken a new full backup. The difference is that this synthetic full backup was created by taking only an incremental backup of the client and having to pull that over the network.

Archiving Archiving is not the same as backing up your data. Archives usually take the form of compliance archives that adhere to regulatory requirements relating to authenticity of data and nonrepudiation. Archives need to be indexed and searchable, provide a secure and reliable audit trail for data, and built on top of reliable media that can be easily refreshed. Archives need to be planned with the very long-term in mind.



Storage Management

TOPICS COVERED IN THIS CHAPTER:

- ✓ **Capacity management**
- ✓ **Chargeback and showback**
- ✓ **Performance management**
- ✓ **Alerting**
- ✓ **Storage Resource Management**
- ✓ **Management protocols and interfaces**

Storage management is a broad topic that covers all things relating to the management and monitoring of your storage estate. The three major areas of management are capacity, performance, and availability.

On the capacity front, the major objectives are simple: make sure you always have enough capacity to service your applications, and make sure that capacity is utilized in the best way possible. Good capacity management also requires simple and clear reporting and trending.

On the performance front, storage management includes ensuring that your storage estate is high performance—performing well enough for your applications and requirements—as well as identifying performance bottlenecks and appropriately tuning performance. A major part of this includes providing performance-related statistics.

On the availability front, storage management is about monitoring and managing service and component-level redundancy, ensuring that storage services remain up and working.

These three areas can be easily summarized as *good storage management is about making sure that the storage is always on (availability), always has space (capacity), and is fast (performance)*. If your estate fits that description, you're doing a good job.

An important rule in good storage management is to be prepared. When everything is going well, people will be happy and civil. However, when the going gets tough and people think they are having performance or availability issues, they quickly start blaming everybody and everything. When this happens, you will need the ability to dig deeper and provide in-depth analysis and evidence.

Good storage management requires solid processes, policies, and tools. An example of a good process is life-cycle management, including commissioning and decommissioning servers and services. If you just turn off servers and unplug them without reclaiming the storage associated with them, you will end up with a hopelessly underutilized storage estate with islands of wasted capacity. On the policy front, having a policy that all changes to production systems must be carried out by two competent staff members might well save you a major outage in your production environment. And on the tooling front, being able to provide charts and other supporting data will be invaluable to you when you're put on the spot over a high-profile performance issue.

In this chapter, you'll delve into all of these topics, plus more.

Capacity Management

First and foremost, capacity management is about making sure you don't run out of space in your environment. After all, providing storage capacity—space to store your files and data—is a bread-and-butter requirement for any storage estate.

However, there's a lot more to capacity management. For example, capacity management in an FC SAN–based storage estate includes having enough of all of the following:

- SAN switches
- SAN ports
- SAN bandwidth
- Inter-switch links (ISLs)
- Front-end array ports
- Data center power and cooling
- Data center floor space
- Cabinet space for more drives
- Licenses for usable capacity

The list could go on. The point is that capacity management includes a lot more than just thinking about gigabytes and terabytes. You'll look a bit foolish if you go through the pain of having a purchase order raised to install 64 TB of additional capacity for one of your storage arrays, only to find that the array is already at full capacity.

Real World Scenario

What Can Happen if You Don't Plan Every Detail of Your Upgrades

One company was happily going about its business, secure in the knowledge that its primary storage array was only half full and could easily be expanded if required. Shortly before using up all existing capacity in the array, the company raised a purchase order for an expansion cabinet and 160 TB of storage to be purchased and installed. This was all approved, and the upgrade was ordered. However, on the weekend when the upgrade was planned to be installed, the installation engineer found that there wasn't enough space on the data center floor to install the new expansion cabinet. It turned out that since the storage array had first been installed, additional server racks had been installed in the data center, and one of these cabinets had been installed a bit too close to the storage array, meaning that there wasn't enough space to bolt the expansion cabinet onto the array.

Fortunately, in this particular instance, the server rack causing the problem had only a single server in it, and the following weekend the server was powered down and the server and rack were moved to another location in the data center. It was not the end of the world, but it was slightly embarrassing and could easily have been far more difficult to resolve.

When planning the layout of this data center, it was never identified or mentioned that the controller cabinet for the storage array was an inch or two wider than a standard data center cabinet. In fact, in the past the storage industry has been a terrible data center citizen and has often rolled out technology in bespoke cabinet sizes, making data center planning a nightmare. Fortunately, most storage arrays these days are starting to ship in standard rack sizes, but it's always worth double-checking!

While there are many aspects to capacity management, the bulk of it deals with available gigabytes and terabytes. So let's take a look at some technologies that influence capacity management and then talk about how we report on capacity management.

Capacity Reporting

In the modern world, IT is often seen as a drain on cost, and as a result, execs and shareholders quite often want to ensure that IT is spending money and resources wisely. That's a good thing. Hopefully, none of us want to waste resources! However, this often requires you to be able to *prove* that you're not wasting money and resources, and one of the best ways to prove this is via fancy reports. Execs love fancy reports as well as raw spreadsheet data to back them up!

When it comes to capacity reporting, there is no better way to justify your existence as an IT department than by showing senior folks within the business where all the money is going. It's even better if the money is being spent on the *business* and not on *IT*. For example, Figure 12.1 is a good report, because it shows that most of the storage estate is being consumed by business units. On the other hand, though, Figure 12.2 isn't such a good report. Figure 12.2 shows IT as a huge consumer of storage resources (and storage resources cost money) in comparison to the front-office line-of-business users.

In Figures 12.1 and 12.2, we've grouped a lot of shared corporate functions together. These functions are services such as corporate email, email archiving, finance systems, HR systems, and other similar services that are shared corporate wide. This is a common practice in keeping reports readable. However, be prepared for management to request that you provide a separate breakdown of what is using up storage in the shared corporate space.

Other examples of capacity reports include breaking down capacity utilization across different zones or environments, such as the following:

- Production
- Development
- Staging
- Research

FIGURE 12.1 IT as a small consumer of capacity

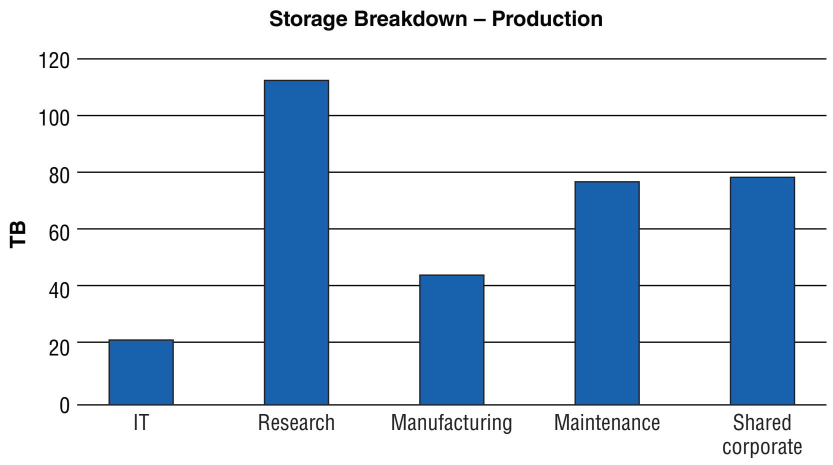


FIGURE 12.2 IT as a large consumer of capacity

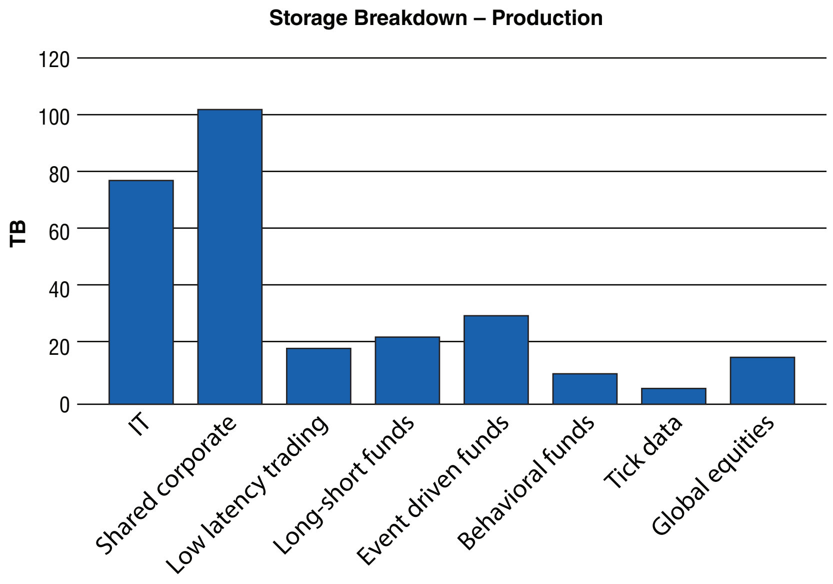


Figure 12.3 shows a breakdown of these environments, with the addition of separating production into production-live and production-DR. This additional breakdown of production-live and production-DR is quite common, as it highlights how much storage is being consumed for disaster recovery (DR) purposes.

FIGURE 12.3 Capacity breakdown by environment

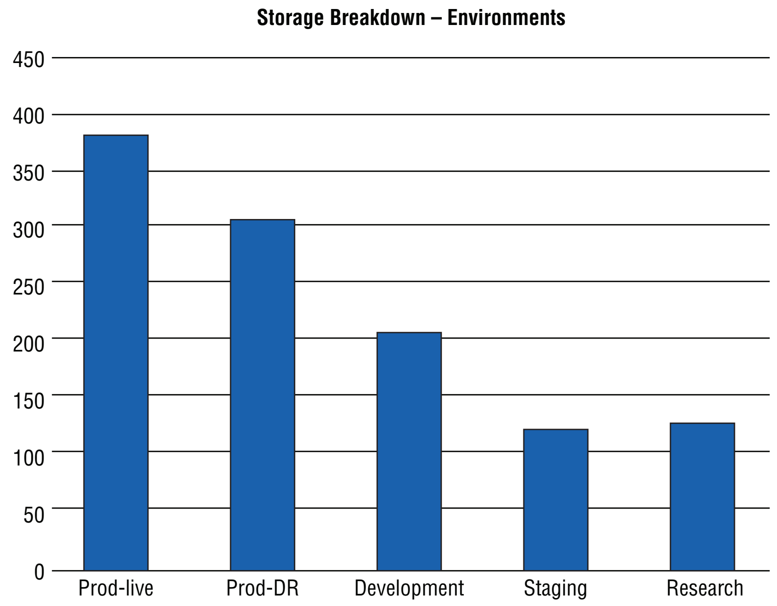


Figure 12.4 shows the same breakdown of environment data represented as a pie chart.

You may also be asked to show data such as net capacity usage year-to-date, so management can see who is consuming storage the fastest this year. Figure 12.5 shows an example of such a report for a company that has several lines of business:

- Food sales
- Homeware sales
- Insurance
- Internet banking

FIGURE 12.4 Breakdown by environment pie chart

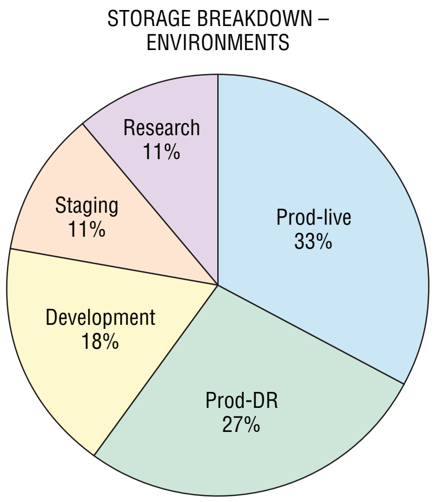
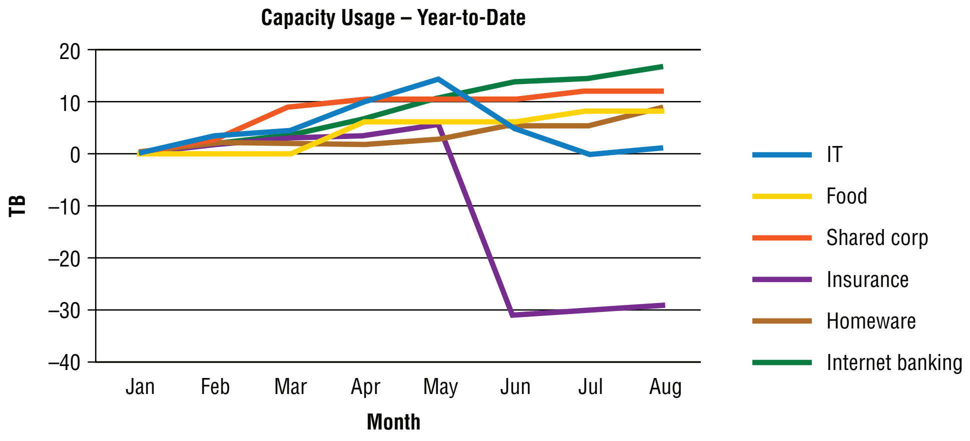


FIGURE 12.5 Net capacity use year-to-date



Thin Provisioning Considerations

Although thin provisioning (TP) is fantastic technology that has huge capacity-related benefits, it makes capacity management a whole world harder! Let's now discuss the challenges that over-provisioning can introduce to capacity management, and the added importance it adds to forecasting and trending.

Overprovisioning

The issue with TP, when it comes to capacity management, is that TP allows us to over-provision arrays. For a detailed discussion of thin provisioning and overprovisioning, see Chapter 10, "Capacity Optimization Technologies." For now, though, let's just say that *overprovisioning* allows us to pretend we have more storage than we really do. That's all well and good, but it introduces huge new risks, and if we do not manage these risks properly, we can lose our jobs. For example, if enough systems call our bluff and demand their entire allocation of storage, we may not have enough capacity to give them. If this happens, systems start crashing.

NOTE

Overprovisioning is sometimes referred to as *oversubscribing* or *overallocating*. The term *overprovisioning* is also used in the context of solid-state memory devices to indicate a device that pretends to the operating system that it has less capacity than it really does. Clearly this is the opposite of the meaning in relation to thin provisioning. On a solid-state device, such as a 400 GB flash memory drive, the device might have 600 GB of actual capacity but be using the hidden 200 GB to perform housekeeping operations and thus increase the working life span of the device. However, when overprovisioning in the context of thin provisioning, we pretend we have more capacity than we actually do.

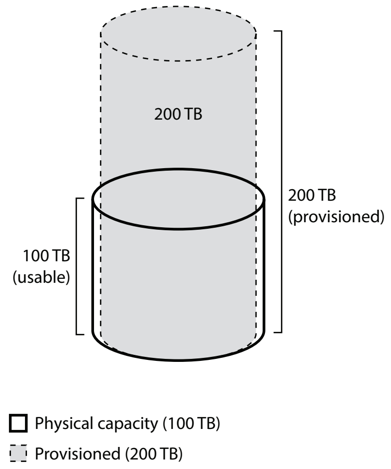
Figure 12.6 shows a logical representation of an overprovisioned array. The array has 100 TB of physical capacity, but is 100 percent overprovisioned, meaning that it is pretending to have 200 TB of capacity. And this is fine while the array is relatively unused, maybe with 50–70 TB of actual used data, because it would take quite a big run on the array (think of a "bank run") to use up that remaining 30 or 50 TB of data. However, if 90–95 TB of the array is used, you're much more likely to run out of capacity soon. So you'd better get rid of some unnecessary data—maybe move it to another system or delete it—or install some more capacity very quickly!

Anyway, thin provisioning and overprovisioning are here to stay, so we may as well tool up so that we can effectively capacity-manage our overprovisioned arrays.

WARNING

It is highly recommended that you get agreement from appropriate senior IT management before going down the road of overprovisioning. This should include an agreement to purchase additional capacity in a predictable amount of time, because of the added risk overprovisioning brings. The last thing you want to do is introduce new risks without getting agreement from those who have overall responsibility for the IT estate and service delivery.

FIGURE 12.6 Logical representation of an overprovisioned array



The Need for Trending

After you have approval to implement overprovisioning, you definitely want to start out slowly. Start out by overprovisioning by a small percentage, maybe 10–20 percent. Then make sure you trend and forecast for a few months; I recommend at least four to six months. Once you have a good idea of the capacity-related characteristics of your estate and your arrays, you might want to push overprovisioning up to something more adventurous, such as 30–40 percent. Keep trending and forecasting, and repeating the cycle. There's no arbitrary limit on how much you should or shouldn't overprovision; it really depends on your environment. The key here is to understand the growth and characteristics of your environment. That being said, overprovisioning too much is *not* recommended!

The key metrics to get to know and love in order to trend and report on your overprovisioned environment are as follows:

Physical Capacity This refers to the usable capacity of the array. Think of this as your approved spending budget; it is safe, secure, and approved for you to spend as you need it. You cannot use more than this. You can *provision* more than this, but you cannot *use* more than this.

Provisioned Capacity This is how much you have allocated to your hosts, or put in other words, *how much you are pretending to have*. When overprovisioning, you pretend to have more than the physical capacity of your array.

Used Capacity This is the amount of capacity your hosts have actually written to.

Each of these metrics must be known and tracked *individually for each array* you have.

Let's glue all of this together in a quick example. We can have an array with 100 TB of *physical capacity*, and if we turn on overprovisioning, we can pretend we have more than that. Let's say, for instance, we provision a grand total of 150 TB to connected hosts. That 150 TB value will be our *provisioned capacity* figure, making our array 50 percent overprovisioned, or sometimes referred to as 150 percent provisioned. But of that 150 TB that we provisioned, let's say our hosts have only written to 60 TB of it. That makes 60 TB our *used capacity* figure. This is summarized as follows:

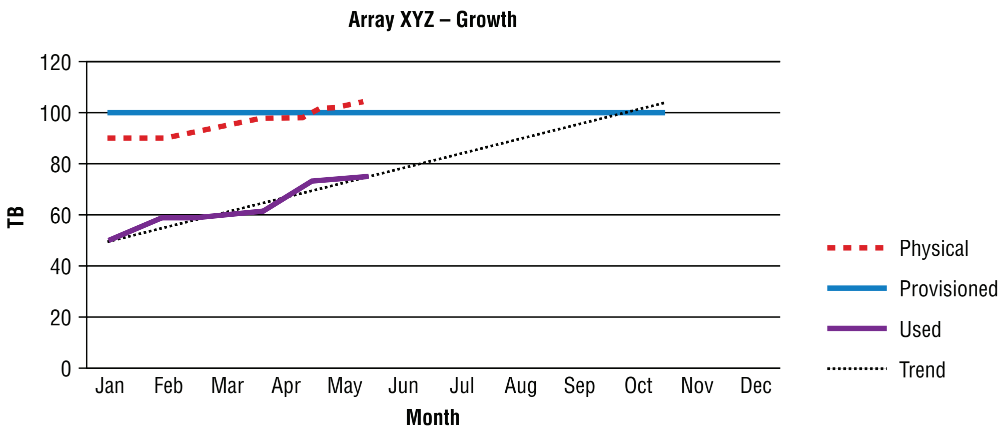
- Physical installed capacity: 100 TB
- Provisioned capacity: 150 TB
- Actual used capacity: 60 TB
- Overprovisioned percentage: 50 percent

Now then, if our used capacity is increasing by 5 TB per month, it will be eight months before our *used capacity* consumes all of our *physical capacity*. If that ever happens, all hell breaks loose! Our hosts still think we have 50 TB of free space, but we don't, because we lied to them about how much space we have. So any new writes that come into our array will fail, and applications and services will start to crash!

So, monitor and trend, and then monitor and trend some more. Make sure you buy more capacity before you run out of available space! An important part of achieving this is understanding your purchasing cycle, as well as your vendor's lead times to deliver a new kit. There's no point in waiting until you have one month's worth of storage left in the bank if it takes you three weeks to get purchase orders approved and your vendor takes four weeks to deliver new equipment!

Figure 12.7 shows the kind of monitoring and trending graph that would be useful in the example just cited.

FIGURE 12.7 Trending overprovisioning



It's also important to keep track of business trends, such as plans to acquire new companies, plans to open new markets, plans to migrate to new applications, and so on. All of these can have a significant impact on capacity planning in the estate. For example, replacing a legacy application with a new application may require you to run the two applications side by side for a short while, during which time your storage requirements may be doubled.

Another good idea, if your environment is small and manageable enough, might be the following: make sure that you always have enough free capacity (physical capacity minus used capacity) on your array to be able to deal with any one server consuming all of its allocated capacity. For example, if your server has 5 TB of allocated capacity, but that server is consuming only 2 TB, it could potentially decide to write to the remaining 3 TB. To protect your array from that scenario, you'd need to have at least 3 TB of free capacity on the array. Less than 3 TB of free capacity in the array would bring the system down.

As a final word on overprovisioning, if there's one thing to monitor in an overprovisioned environment, it's actual capacity used against actual capacity installed!

Deduplication and Compression

Deduplication and compression are popular capacity optimization technologies covered at length in Chapter 10. Both technologies deal with storing more data in less space.

Tracking how effective these technologies are can be useful for internal IT reporting. However, it's unlikely that management outside IT will care about the effectiveness of technologies such as deduplication and compression.

Deduplication effectiveness is expressed as a ratio, with a couple of examples listed here:

- 4:1
- 10:1

A deduplication ratio of 10:1 is better than 4:1. 10:1 tells us that we have stored ten times as much data as we would have if we weren't using deduplication, whereas 4:1 tells us we have stored only four times as much information as we would if we hadn't had deduplication. These ratios can also be expressed nicely in graph format, as shown in Figure 12.8.

In Figure 12.8, we express deduplication efficiency as a percentage rather than a ratio. A problem with ratios is that they can look misleading. For example, the difference between a ratio of 2:1 and a ratio of 3:1 is a whopping 16 percent. The difference between a ratio of 10:1 and 100:1 is only 9 percent, and the difference between 100:1 and 500:1 is a difference of only 0.8 percent. So we get massively diminishing returns as the ratio numbers get higher.

NOTE We convert deduplication ratios to *percentage space saved* as follows:

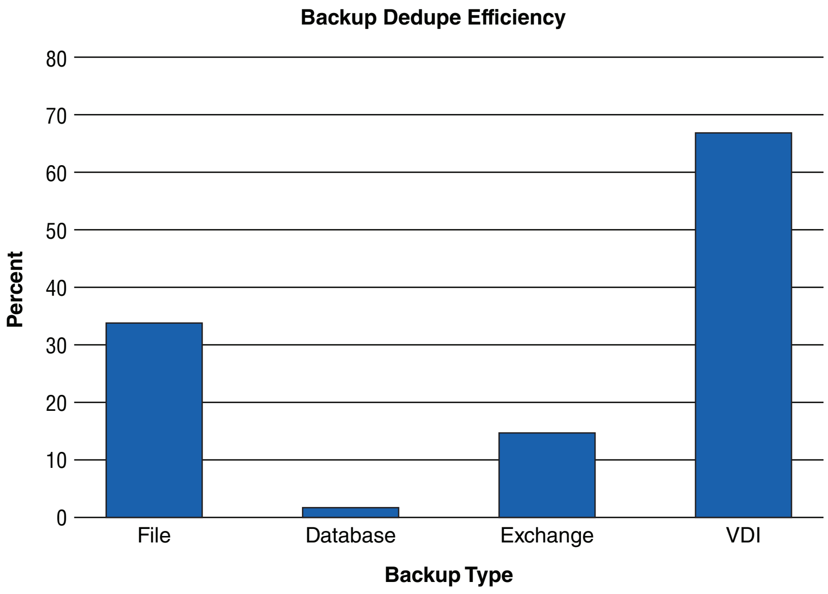
10:1 is converted as 100 – (1/10 × 100) = 90 percent

25:1 is converted as 100 – (1/25 × 100) = 96 percent

2:1 is converted as 100 – (1/2 × 100) = 50 percent

3:1 is converted as 100 – (1/3 × 100) = about 66 percent

FIGURE 12.8 Deduplication efficiency



Be aware that in things like data reduction, seemingly small differences can have a large impact, especially on larger systems. For example, a vendor may argue that a system that achieves 95 percent reduction is only 5 percent better than their system that gets 90 percent reduction—and 5 percent might seem a like a small number. But on a 100 TB data set, the better system will need only 5 TB of capacity to store the data, whereas the other will require 10 TB.

From a capacity management perspective, there is a big difference between inline and post-process deduplication technologies. Post-process deduplication can require you to have enough capacity to be able to stage incoming data to. This incoming data is non-deduplicated. But the data will be deduplicated later, and after this happens, you won't need that space anymore. For example, assume that each weekend your backup jobs back up 100 TB of data and send it to a deduplication appliance that performs port-process deduplication. So, in the first place, you will need at least 100 TB of free space on the dedupe appliance to store the incoming back-ups. The backups sit there in their non-deduplicated form until a later time, when the dedupe appliance runs a deduplication job and reduces the data set down to, let's say, 30 TB. So in the first place, you needed at least 100 TB of space, but after it's deduped, you needed only 30 TB. That is pretty wasteful. For this reason, inline deduplication is a preferred method.

NOTE Some post-process deduplication technologies work on a schedule basis, where you schedule a quiet time for the deduplication operation to occur. A popular example was the post-process deduplication on a NetApp FAS technology (primary storage array rather than a backup target); it was common practice for deduplication jobs to be scheduled for the early hours of the morning. However, other post-process deduplication technologies start the deduplication process much sooner, meaning that the deduplication is occurring at the same time as the data is being ingested. In the latter case, you don't need the large landing pad, but ingestion rates are hampered by the deduplication processing.

Quotas and Archiving

In the world of files and emails, quotas can be used to rein in and enforce limits on the amount of storage space users and departments consume. It's a fact of life that if you don't restrict the amount of storage people have access to, they'll keep asking for more and more until there isn't any left. However, be aware that limiting the amount of storage users have for storing their work will often encourage them to develop bad practices such as storing documents outside the corporate network in unauthorized locations such as the public cloud and USB sticks. All of this moves that data outside the control of corporate governance, which includes backup, archiving, antivirus, and security. So beware of this when considering the use of quotas.

It's common to implement various levels of quotas for email. For example, a three-tiered approach to email quotas is popular and might work like this:

Quota level 1: When a user's mailbox usage reaches this quota level, the user will receive daily emails informing them that they have reached their quota limit.

Quota level 2: When this quota level is reached, the system will stop sending emails from the user, and they will receive an email informing them of this.

Quota level 3: When this quota level is reached, the system will stop receiving emails for the user.

This kind of three-tiered approach to quota management gives users plenty of opportunities to tidy up their mailboxes before the system stops sending and receiving mail for them.

However, many organizations consider it too dangerous for their employees to be forced to delete their emails. There's also an associated burden on IT whenever a user deletes the wrong email; it's usually IT that has to recover it. Add to this the fact that senior management and company execs will almost certainly be exempt from such quotas, and it becomes evident that a better solution is needed.

That's where things like email archiving come into play. For example, email archiving can be configured to silently move emails that are older than a certain age to an archive. This archive can be on cheaper storage and usually exists outside the normal corporate backup schedule, making it an ideal place to store older emails. And critically, as far as the user is concerned, the email is still in their inbox; it may just have a slightly different icon or take a couple of seconds longer to open. Aside from that, the mail is still in their inbox as far as they're concerned.

These kinds of solutions can do away with the need to manually delete or manually archive old emails, and they give users the impression that their inbox is bigger than it actually is. It also enables IT to save money by placing archived emails and documents on cheaper storage that doesn't burden the backup infrastructure either. Everybody wins.

The same principles apply to file shares and SharePoint. If you don't enforce quotas, these file shares and SharePoint sites will become dumping grounds with uncontrolled growth. And when quotas fail—maybe senior management doesn't like them—then archiving technologies and Information Lifecycle Management (ILM) practices come into play. Policies such as the following can be defined:

- Files not modified in the last *x* months can be moved to lower tiers.
- Files not accessed in the last *y* months can be moved to lower tiers.
- Files owned by a particular department are placed in a compliance archive.
- Files in a certain filesystem are compressed.
- Files of a certain type are stored for a specified period of time.

Policies like these tend to work on the basic principle that new and frequently accessed data belongs on high-performance, high-cost storage, and data that is infrequently accessed belongs on lower-tier, cheaper storage. However, ILM is more complex than this, as are attempts to classify data according to its business value, place it on appropriate infrastructure, and apply appropriate policies to it throughout the entire life of the information—from creation to deletion. As you can imagine, this is extremely difficult to do.

NOTE It's important to understand that Information Lifecycle Management (ILM) isn't a hardware or software product. Instead, it's a strategy that incorporates policies and procedures that are constantly evolving. It's also a bit of a marketing buzzword. ILM is one of the IT nirvanas that nobody is anywhere near reaching yet.

Does this sound a bit wishy-washy? It is! In the real world, ILM is a challenge, and most organizations have immature ILM strategies. To be honest, at most organizations, ILM is one of the lowest priorities, if it's even a priority at all! However, one area in which some ILM principles are well adhered to is compliance archiving. Compliance archiving places certain data sets and data types into archives, where they are protected from tampering and deletion for a strict period of time determined by legislation. However, compliance archiving is only a tiny part of the overall ILM picture.

Showback and Chargeback

Showback and chargeback are common practices in many companies. They relate to highlighting costs and billing costs, respectively. *Showback* identifies and highlights the costs of IT services being consumed by a certain customer or business unit, but does not actually charge them. It just shows the costs. C*hargeback* takes it one step further and includes a bill!

Both have their pros and cons. Obviously, the fact that chargeback includes a bill has a bigger impact on behavior. For example, departments are less likely to ask for more storage than they need if they know they'll have to pay for it. On the downside, though, chargeback can encourage some bad behaviors and practices. For example, in order to avoid costs, departments might ask for lower-performance (cheaper) storage than they need. Or they may ask for nonreplicated storage, because it costs less than replicated storage, when in reality they need replication. They may also start storing data outside the authorized corporate IT system, in places such as the public cloud. So it's a balancing act.

Showback, on the other hand, isn't as much of a deterrent, as no money changes hands. However, if the showback reports are reviewed by senior management, it can result in senior management applying pressure to departments to justify or curb their use of storage.

Real World Scenario

An Example of Internal Chargeback Leading to Bad Practices

A legendary story at one company I worked at told of a development system that was built cheaply in order to avoid a large internal chargeback bill. After all, the system was only a development system and didn't need high-performance, bulletproof storage. So the development system was built on low-tier storage that wasn't replicated. Then, as is so often the case with development systems, overnight and without anyone particularly noticing, it became an important live production system! It had business units relying on it to do their core trading.

At this particular company, it was policy for *all* volumes on production systems to be replicated. However, this system wasn't. Instead, parts of it were being copied to a second system at the DR site, but this copy was a simple rsync copy, and the target for the rsync job was a random server that the developers had some spare capacity on. This is absolutely not how you'd want a production trading system to be built.

Fortunately, in this case, no major incident occurred, but as more and more business units started relying on this system, its performance quickly started to drop. Upon investigation, it was discovered that the system was on the lowest-performance storage and not being replicated.

If used properly, showback and chargeback can be useful. Both can encourage more-responsible behavior when it comes to consumption of expensive storage resources. A department is more likely to seriously consider its storage requirements if it knows it has to pay for them. Both kinds of reports can show a paper trail of where the money that gets pumped into IT actually goes; both reports show who's using what. So if an IT department is ever asked to show where it's spending all the money, these reports can show that business unit *x* is consuming *y* storage at a cost of *z*.

Performance Management

Performance management is about ensuring the optimal performance of the entire storage estate. It includes arrays, the network, NICs and HBAs, and even host-based components such as volume managers and multipathing software. It should encompass the entire picture of the storage estate.

NOTE It's important to note that it's entirely possible for performance of an application to be so bad that the application is considered down. Imagine a database server that's so slow that queries from web servers time out. The database server might still technically be *up*, but as far as customers trying to buy something through the website are concerned, the site is down and they can't buy. So performance is vital.

While there seems to be no end to the list of things to monitor, a vital concept is baselining.

Baselining

When it comes to troubleshooting a suspected performance issue, it's vital to have a frame of reference, something to compare against. This is where *baselining* comes into play.

NOTE Some performance management tools have the ability to hold a database of historical performance data that can be recalled and compared against in the event of a performance issue. Even so, it's still highly recommended to maintain additional baseline data outside the performance management tools database—for example, in document formats such as Excel spreadsheets. A major reason for this is that performance management tools can lose their historical data. Events such as upgrades to the performance management software have been known to lose data or perform such major updates to the backend database schema that the old data can no longer be used in the new version of the tool (requiring a separate instance of the old version to be used just in case you need to read historical performance data). That is not ideal. It's also possible that the performance management tool be so slow to chart historical data that it's practically useless in the heat of a performance issue. For these reasons, it's highly recommended to keep separate copies of baseline data.

There's nothing worse than being in the thick of a performance issue, looking at your real-time performance stats, and thinking to yourself, *is that good or bad?* That's exactly what can happen if all you have to go on is how things currently look. How are you supposed to know if 15,000 IOPS with a response time of about 15 ms is good or bad? For some applications it might be fine, and for others it might not. It's all well and good having pretty-looking charts and graphs, but when you're asked whether the data they show is good or bad, and you can't answer with confidence, you're going to look a bit daft. What you need is a copy of the same stats at a time when there was no perceived performance issue. That way, you can compare the two and see if there is a difference.

For example, say your current stats show 12,000 IOPS at a response time of about 25 ms for the volume that is *supposedly* performing poorly. If you can compare the stats to the same day a week ago or a month ago, and the stats from then were also approximately 12,000 IOPS and around a 25 ms response time, you have a good set of facts that suggest the problem is not storage related. However, if the stats for the same volume from a week or month ago show 40,000 IOPS with an average response time of about 5 ms, or maybe 8,000 IOPS with a 5 ms response time, then you know there might be something happening with the storage.

As you can see, performance baselining can be a lifesaver. It's a good practice to regularly re-baseline your storage environment, maybe on a quarterly basis, and store the data for future reference. If you don't take regular baselines, you at least need to take them when any major changes are made to the environment—and major changes include things like hardware and software upgrades, as well as system migrations. In fact, when major changes are made to the environment, you should baseline before and after and then compare. The last thing you want is an application owner saying he's getting only half the normal performance since you upgraded the environment. If you have no stats to prove otherwise, you're in a difficult position.

NOTE

Baselining before a major infrastructure change is what is considered a best practice (BP). There are loads of best practices that you should stick to as part of good storage management. Many of these BPs, especially those that are technology specific, are driven by your vendors. However, some BPs apply no matter what the technology is. For example, making a backup of your existing configuration before making a major change is a good practice regardless of whether we're talking about switch firmware upgrades or server migrations. Other best practices might also include rules such as not deleting LUNs during business hours or not making changes to trading-related systems during trading hours. At the end of the day, best practices are designed to make your life easier and keep systems up and running in an optimal way. So it's a good idea to maintain and regularly review a list of best practices.

Latency/Response Time

Probably the biggest performance killer in a storage environment is latency (sometimes called response time). *Response time* is the time it takes to send a command and to get the response. Any delay imposed on response time is referred to as *latency*.

High latency causes irate users, and if it gets high enough, applications can become useless to the point that they are considered down. It's probably fair to say that in a storage environment, the only things worse than high latency are data loss (DL) and data unavailability (DU). So take latency seriously.

Latency can occur at just about every layer in the design. Here are the common places, although the list is not comprehensive:

Host: Filesystem, volume manager, host bus adapter (HBA)

Network: SAN or IP network, inter-switch links

Storage array: Array front-end ports, cache, backend, replication

It's a common misconception that high IOPS means low latency. After all, IOPS is an acronym for input/output operations per *second*, and the reference to time makes people think that IOPS and latency are linked. In some respects they are, but it is vital to understand that the two metrics do not scale linearly. That is, if a system can do 10,000 IOPS at 1 ms latency and also 20,000 IOPS at 2 ms latency, this is by no means saying it can do 100,000 IOPS at 10 ms latency and 200,000 IOPS at 20 ms latency. In fact, it almost certainly won't be able to do the latter. IOPS and latency do not scale linearly! The following may help clarify: If your storage array could process only one request at a time, 1,000 IOPS would mean 1 ms latency. However, in the real world, storage arrays can process multiple requests in parallel, meaning that IOPS represents the aggregate performance across the entire system, whereas latency tells you how quickly the array can process any single request.

There's no point in looking at an array spec that lists only IOPS. A figure of 250,000 IOPS means *absolutely nothing* without at least having a latency figure associated with it. 250,000 IOPS at 10 ms latency is a totally different application and user experience than 250,000 IOPS at 50 ms latency. A useful (but by no means perfect) example, is an order-processing department that can process 100 orders per day. But processing 100 orders per day, as well as shipping goods the same day by express courier, is an entirely different customer experience than processing 100 orders per day but having nobody in the postal room except for Mondays, and then sending goods out by the cheapest, slowest mail service possible. Both can do 100 orders per day, but one is clearly better than the other.

In a Fibre Channel–based SAN environment, the biggest and most common contributor to latency is usually the spinning disk drive. This is due to its mechanical nature in a world dominated by silicon and electrical components. FC switching, on the other hand, imposes very little latency. This is because FC switches usually perform cut-through switching; the switch frame pretty much starts leaving the switch (egress) before the entire frame has entered the switch. It's on its way out before it has even arrived! Cut-through switching is faster than store-and-forward switching, in which the frame has to be entirely buffered on the switch before being passed on to the next hop in its journey. As a result, FC switching usually adds very little latency—usually in the low microseconds, though some vendor documentation lists it in nanoseconds. Either way, it is minuscule compared to the latency incurred by spinning disk, which is measured in milliseconds. For example, spec sheet numbers for a 3 TB Seagate Barracuda drive list read and write latency as follows:

- < 8.5 ms average read performance
- < 9.5 ms average write performance

And you can bet those numbers are as best-case as possible.

Obviously, solid-state media, such as flash memory, can be used to replace a lot of spinning disks and reduce a lot of the mechanical latency common to spinning disks.

In a NAS environment operating on IP over Ethernet networks rather than FC networks, network latency can be much more of a factor. That's not just because Ethernet switches commonly deploy the slower store-and-forward switching method. It's also because IP and Ethernet networks experience a far higher rate of packet drops requiring retransmission of packets. All in all, network latency in NAS and sometimes iSCSI environments is more of a factor in the overall end-to-end latency.

Some other things that can increase latency in a storage environment include the following:

- Random I/O
- VMware

VMware, as well as other hypervisors, is known for creating what the industry calls the *I/O blender effect*. This occurs when all the discrete I/O streams coming from each virtual machine on a single physical server are all mixed and munged together by the hypervisor and hit the network as a truly randomized mess of I/O. This I/O is hard to service quickly, especially by traditional spinning disk, and results in increased latency. One of the main reasons is that it screws up the array's attempts to identify patterns in the data and prefetch data into cache. It's a little bit like a hundred people speaking at the same time. Although they all might be having perfectly reasonable conversations, the sheer number of them means that it's incredibly hard to isolate individual voices and make sense of what they are saying. Fortunately, solid-state media can help here.

Needs for Very Low Latency

There are plenty of real-world requirements where ultra-low latency is of paramount importance. Some of those use cases include the following:

- Low-latency trading systems
- Online shopping systems that have to perform fraud detection before processing payments
- Touch-in travel-card systems such as the London Oyster card, requiring commuters to touch the card on a panel to gain access to subways via a turnstile, and card balances have to be verified before a commuter can pass through the barrier
- Pay-as-you-go cell-phone systems requiring account balances to be checked before connecting calls

IOPS

IOPS, another well-known and popular storage performance metric, is commonly used to measure random workloads. It's an acronym for input/output operations per second, and it is a measure of how much work a system is doing, or *can* do, in a second.

However, the concept of IOPS is vague and abused, making it pretty useless on its own. What exactly is an I/O, and are all I/Os equal? All I/Os are *not* equal. Some are big; some are small. Some are read; some are write. Some are random, and some are sequential. Most real-world workloads contain a mixture of them all: small, large, read, write, random, and sequential.

An IOPS number without an associated latency number is absolutely useless. Who cares if an array can do 500,000 IOPS, if the latency when doing those 500,000 IOPS is over 50 ms? An array that can do 500,000 IOPS at sub-10 ms latency is certainly more desirable than an array that does 500,000 IOPS at 50 ms latency.

More often than not, when vendors quote you their dazzling IOPS numbers, they almost never tell you anything about the type of I/O or the associated latency. In fact, some vendors have been known to quote ludicrous IOPS numbers (well into the millions), and when quizzed on how they were achieved, were forced to admit that they were tiny read I/Os that were answered from buffers on the array's front-end ports—an unbelievably unrealistic and ridiculous scenario. Not only were the I/Os not being serviced from backend disk, they weren't even being serviced from DRAM cache. All I/Os were serviced from local buffers on the array ports themselves. As if that would ever happen in the real world! To be fair to the vendors, often their senior technology people oppose the publication of the numbers on the marketing materials, but are overruled by the marketing department.

In conclusion, if you are looking to buy a new array or are investigating a potential performance issue, you need to know details about the IOPS, such as size and read/write ratio, as well as the latency associated with those I/Os. If all you have is an IOPS number, it's utterly meaningless.

MBps and Transfer Rate

MBps is another of the popularly quoted storage performance metrics. It's an acronym for *megabytes per second* that refers to the number of megabytes a disk drive or storage array can transfer in a second. It's commonly used to measure or express the performance of sequential (nonrandom) workloads such as backup and restore jobs, as well as things like streaming media.

NOTE It's important as well to note that MBps refers to mega*bytes* per second and not mega*bits* per second. Mbps, or Mb/sec, refers to megabits per second.

Technically speaking, transfer rates are measured in MBps¸ and the two terms are often used interchangeably when referring to the performance of disk drives and solid-state drives. However, more often than not, when referring to the throughput of a storage array or an entire solution (end-to-end), we tend to use only the term MBps.

It's important to note that throughput-driven applications, like the backup and streaming media examples already discussed, aren't nearly as bothered about latency as predominantly random-driven workloads such as transactional databases.

Factors Affecting Storage Performance

Many things can have an impact on storage performance. Several come up time and time again. So let's take a look at some of the more common ones.

RAID

Although RAID can be our best friend when it comes to protecting our data from loss and unavailability, it can bite us on the performance front. What it gives with one hand, it often takes away with the other.

The RAID levels that usually cause the most concern from a performance perspective are RAID 5 and RAID 6, and it's not uncommon for database administrators (DBAs) to demand SAN volumes that are not RAID 5 or RAID 6.

The RAID Debate

In 2003, a couple of DBAs, James Morle and Mogens Nørgaard, launched an initiative called Battle Against Any RAID Five (BAARF, `www.baarf.com`) as a way to vent their frustration over the performance impact of RAID 5 on databases. From their enlightened database-dude perspective, they were denouncing any vendor implementation of RAID 5, and for that matter RAID 4 and RAID 3 (and because RAID 3 doesn't fit with their acronym, they ingeniously refer to it as *RAID free* so the BAARF acronym works).

Their attitude toward RAID 5 can be seen on their website, with quotes and statement such as these:

> Enough is enough … James Morle and others have written books where they discussed the *uselessness* of RAID-F [free, four, five] stuff … we've decided to lower our blood pressure … by permanently refusing to have any more arguments about it.

Although a little extreme and close-minded, many DBAs have similar opinions about RAID 5 and RAID 6 technologies. (RAID 6 wasn't popular at the time these two pillars of the technology world issued their authoritative decree.)

However, when all is said and done, RAID 1 and RAID 10 technologies tend to cost a lot more money to implement than RAID 5 and RAID 6. So demanding RAID 1 and RAID 10 is fine, as long as the additional cost can be justified.

Parity-based RAID schemes, such as RAID 5 and RAID 6, perform differently than other RAID schemes such as RAID 1 and RAID 10. This is due to a phenomenon known as the *write penalty*. This can lead to lower performance, especially in cases with workloads that consist of a lot of random write activity—as is often the case with database workloads. The reason the write penalty occurs is that small-block writes require a lot of parity recalculation, resulting in additional I/O on the backend. We discuss RAID and RAID performance in more detail in Chapter 4, "RAID: Keeping Your Data Safe," but as we're here, a quick analogy might help.

Think of writing data to a RAID 5 or RAID 6 volume as being similar to editing a document in Microsoft Word. Imagine that in a Word document, each *sentence* is equivalent to a RAID *stripe*. If we need to replace an entire sentence with a new sentence, that's easy: we just highlight the entire sentence, and then paste the new sentence right over the top of it, deleting the old highlighted sentence at the same time. That's like large-block writes, where we write an entire new RAID stripe over the top of an old one, nice and easy. Now let's say we need to make a few modifications within an existing sentence. Maybe we need to delete the third word, add a comma after the sixth word, swap the seventh and eight word around, and add a dash between words nine and ten, and finally correct a spelling mistake in word ten. This is a whole lot more effort than just pasting a new sentence over an old one. It's the same for updating an existing RAID stripe. Small-block writes are relatively hard work for RAID 5 and RAID 6 (as it is for us making changes within sentences) because they require changes to be made within RAID stripes, which forces the system to read the other members of the stripe to be able to recompute the parity. In addition, random small-block write workloads require the R/W heads on the disk to move all over the platter surface, resulting in high seek times. The net result is that lots of small-block random writes with RAID 5 and RAID 6 can be slow. Even so, techniques such as redirect-on-write or write-anywhere filesystems and large caches can go a long way to masking and mitigating this penalty.

Although it's true that in an ideal world we'd probably all have RAID 10 for everything—mirroring plus striping—it's not always that simple. The main problem is that RAID 1 and RAID 10 consume massive amounts of capacity as they're based on mirroring. This additional required space pushes costs up. As long as you have the data center floor space and the cash in your budget, you can have as much RAID 1 or RAID 10 as you like. If, on the other hand, you need to keep your costs under control, RAID 5 and RAID 6 will probably be better choices in some use cases.

Cache

Cache is the magic ingredient that has just about allowed storage arrays based on spinning disk to keep up to speed with the rest of the technology in the data center. If you take DRAM caches and caching algorithms out of the picture, spinning disk–based storage arrays practically grind to a halt. They'd be so painfully slow that the world would be a different place. We simply wouldn't be able to do many of the things we do today without caching, or at least we wouldn't be able to do them as well as we do them today. Think of driving from Chicago to California without Interstate 80 (I-80) as being a disk-based storage array without cache. Adding I-80 would then be like adding cache (making the journey faster). To extend the analogy, all-flash arrays would be like taking an airplane.

Having enough cache in your system is important in order to speed up average response times. If a read or write I/O can be satisfied from cache (not having to rely on the disks to complete the read or write I/O), it will be amazingly faster than if it has to rely on the disks on the backend. However, not all workloads benefit equally from having a cache in front of spinning disks. Some workloads result in a high *cache-hit rate*, whereas other don't. A *cache hit* occurs when I/O can be serviced from cache, whereas a *cache miss* requires access to the backend disks. Even with a large cache in front of your slow spinning disks, there will be some I/Os that result in cache misses and require use of the disks on the backend. These cache-miss I/Os result in far slower response times than cache hits, meaning that the variance (spread between fastest and slowest response times) can be huge, such as from about 2 ms all the way up to about 100 ms. This is in stark contrast to all-flash arrays, where the variance is usually very small.

Most vendors will have standard ratios of disk capacity to cache capacity, meaning that you don't need to worry so much about how much cache to put in a system. However, these vendor approaches are one-size-fits-all approaches and may need tuning to your specific requirements. It's often a good idea to consult with your vendor when doing this.

Thin LUNs

Thin LUNs (based on thin provisioning) work on the concept of allocating space to LUNs and volumes on demand. So on day one when you create a LUN, it has no physical space allocated to it. Only as users and applications write to it is capacity allocated. This allocate-on-demand model can have an impact in two ways:

- The allocate-on-demand process can add latency.
- The allocate-on-demand process can result in a fragmented backend layout.

The allocate-on-demand process can theoretically add a small delay to the write process because the system has to identify free extents and allocate them to a volume each time a write request comes into a new area on a thin LUN. However, most solutions are optimized to minimize this impact.

Probably of more concern is the potential for some thin LUNS to end up with heavily fragmented backend layout because of the pseudo-random nature in which space is allocated to them. This can be particularly noticeable in applications with heavily sequential workloads. If users suspect a performance issue because of the use of thin LUNs, perform representative testing on thin LUNs and thick LUNs and compare the results.

Network Hops

Within the network, FC SAN, or IP, the number of switches that traffic has to traverse has an impact on response time. Hopping across more switches and routers adds latency, often referred to as *network-induced latency*. This latency is generally higher in IP/Ethernet networks where store-and-forward switching techniques are used, in addition to having the increased potential for traffic-crossing routers.

Multipathing

Many people think of multipath I/O (MPIO) solutions as being all about high availability; if one path fails, another takes over without the application or user even noticing. And that's true. However, MPIO can also have a significant impact on performance.

For example, balancing all I/O from a host over two HBAs and HBA ports can provide more bandwidth than sending all I/O over a single port. It also makes the queues and CPU processing power of both HBAs available. MPIO can also be used to balance I/O across multiple ports on the storage array too. Instead of sending all host I/O to just two ports on a storage array, MPIO can be used to balance the I/O from a single host over multiple array ports, for example, eight ports. This can significantly help to avoid hot spots on the array's front-end ports, similar to the way that wide-striping avoids hot spots on the array's backend.

Standard Performance Tools

Now let's take a look at some of the common performance tools used in the industry.

Perfmon

Perfmon is a Windows tool that allows you to monitor an extremely wide variety of host-based performance counters. From a storage perspective, these counters can be extremely useful, as they give you the picture as viewed from the host. For example, latency experienced from the host will be end-to-end latency, meaning that it will include host-based, network-based, and array-based latency. However, it will give you only a single figure, and it won't break the overall latency down to host-induced latency, network-induced latency, and array-induced latency.

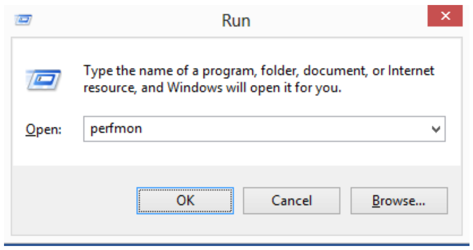
Exercise 12.1 looks at measuring I/O latency by using Windows perfmon.

EXERCISE 12.1

Using Perfmon

In this example, you will use the Windows perfmon tool to monitor the average read and write latency of a SAN volume (D:) on your system.

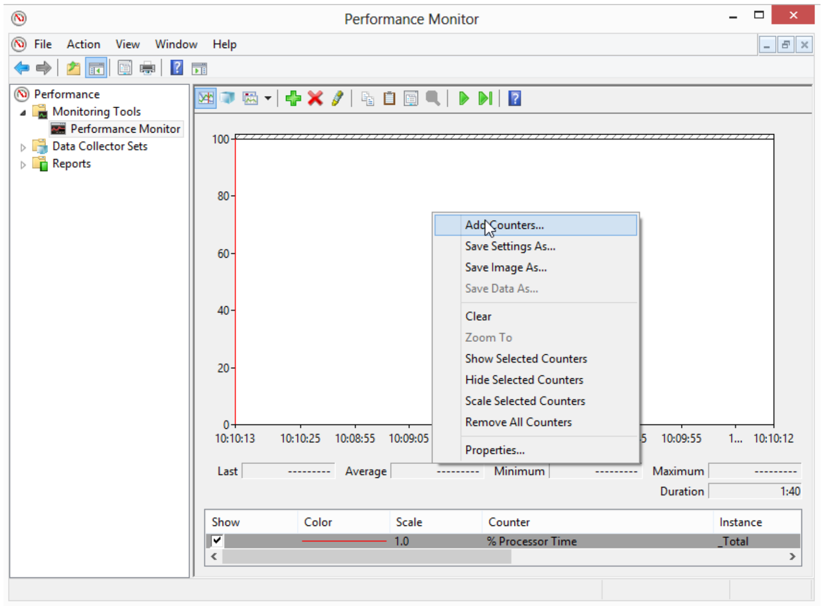
1. Open the Windows perfmon utility by typing **perfmon** at the command prompt of the Run dialog box.


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

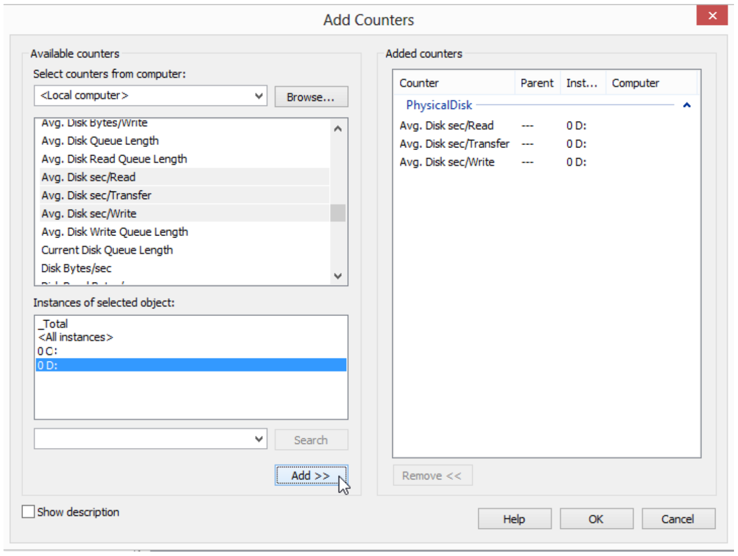
Open: perfmon

2. From within the perfmon window, right-click in the empty white graph area on the right-hand side and click Add Counters.



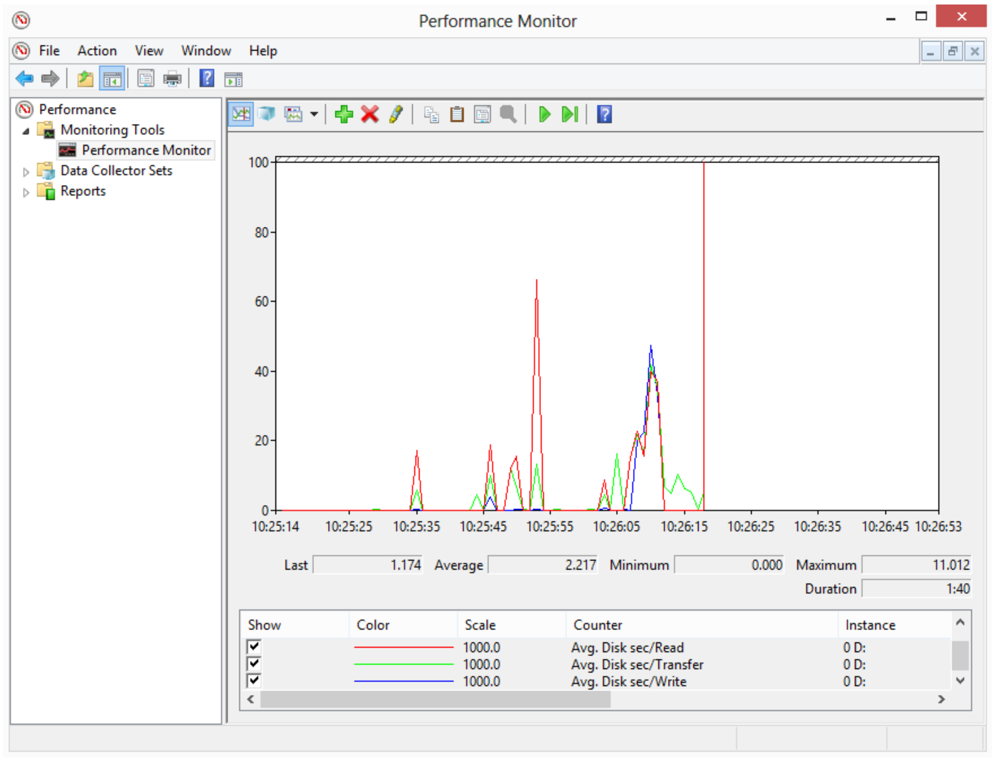
Show	Color	Scale	Counter	Instance
✓		1.0	% Processor Time	_Total

3. From the Add Counters screen, select the following counters to add to the selection: Avg. Disk sec/Read, Avg Disk sec/Write, and Avg Disk sec/Transfer. For each of these, select your SAN volume, volume D:.



Counter	Parent	Inst...	Computer
PhysicalDisk			
Avg. Disk sec/Read	---	0 D:	
Avg. Disk sec/Transfer	---	0 D:	
Avg. Disk sec/Write	---	0 D:	

Now that the counters are added, perfmon will display real-time latency numbers for average read, average write, and average combined read/write latency.



Last	Average	Minimum	Maximum
1.174	2.217	0.000	11.012

Duration 1:40

Show	Color	Scale	Counter	Instance
✓	🟥	1000.0	Avg. Disk sec/Read	0 D:
✓	🟩	1000.0	Avg. Disk sec/Transfer	0 D:
✓	🟦	1000.0	Avg. Disk sec/Write	0 D:

Perfmon can also be used to show many more performance-related counters. Even some of the non-storage-related counters can be useful in troubleshooting suspected storage-related issues. As an oversimplified example, if the CPU busy counter is extremely high on a server, overall server performance will be negatively affected, which will obviously have an impact on the server's storage performance.

NOTE

Perfmon (`perfmon.exe`) replaced the older sysmon (`sysmon.exe`) that was used on some very old versions of Windows.

iostat

Iostat is a common tool used in the Linux world to monitor storage performance. Exercise 12.2 provides a look at the kind of output iostat gives.

EXERCISE 12.2

Using iostat

From the command line on a Linux server that has iostat installed, type the following command to display the I/O stats of that computer every 10 seconds (the figured returned is averaged over the last 10 seconds):

```
iostat -x 10
```

The following output shows the I/O statistics over the last 10 seconds:

```
nigelpoulton@ubuntu-02:~$ iostat -x 10
Linux 3.8.0-19-generic (ubuntu-02)      21/09/13        _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.18    0.00    0.13    1.41    0.00   98.28

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await
    r_await w_await  svctm  %util
sda              12.49     4.27    8.77    2.19   298.69    99.02    72.60
1.39  126.62
    135.39   91.54   4.46   4.89
dm-0              0.00     0.00   11.54    6.44   261.18    99.00    40.05
2.10  116.80
    147.49   61.81   2.57   4.62
dm-1              0.00     0.00    0.36    0.00     1.42     0.00     8.00
0.00    7.59
      7.59    0.00   5.37   0.19
```

While the output to the command may not format well in this book, from the command line on a Linux server, it is easy to read and provides good statistics.

Alerting

It's possible, for even the best of us, to occasionally take our eye off the ball. When this happens, *alerting* can come to our rescue.

At a high level, alerting exists to notify us of important events. These events can be failures, recovery from failures, passing through a capacity threshold, dropping below a performance threshold, or other significant occurrences. Staying at a high level, alerting works on a principle of thresholds and conditions. You configure certain thresholds and associate them with actions to be taken when that threshold is breached. Here are some examples:

- If free space <= 80 percent, send an email alert.
- If any physical component fails, send an email alert.
- If network latency >= 10 ms, send an SNMP trap to the network operations center.

Popular forms of alerting include the following:

- Email
- SMS/text message
- SNMP trap

It is also common with some storage technologies for the system to call home when certain events occur. *Calling home* occurs when the system sends a message back to the vendor, containing information about the condition. Examples include the following:

- The amount of free space has dropped below a threshold.
- Components have failed.

Calling home used to require things like modems attached to the back of arrays and switches. Nowadays, though, calling home is usually accomplished via the Internet using HTTPS. Significantly, only system-related information is sent home; no user data is ever sent back to the vendor.

Depending on your technology, alerting can be configured for a wide variety of conditions. On the capacity front, some of the more useful conditions include these:

- Free space available
- Total used space

The main use of alerting is to make the relevant people (that almost always includes you) aware of certain conditions that need to be addressed. Knowledge is power! There aren't many things more embarrassing than management finding out about a situation in the storage environment before you do. This is far less likely to happen if you have good alerting configured.

More-modern systems call home and report on all kinds of analytical data, and can even identify systems that will be affected by known bugs and alert the user.

Exercise 12.3 walks you through how to configure an alert.

EXERCISE 12.3

Configuring an Alert

On most storage systems, configuring alerting should be simple. The following process walks you through configuring email alerting for failed components on a Brocade switch. To keep the process simple, we'll assume that the switch has already been configured with an email server. You will simply configure email alerting for failed field replaceable units (FRUs) so that any time an FRU fails, an email will be sent to you.

1. From the switch command line, type the `fwmailcfg` command:

```
LegendarySw01:admin> fwmailcfg
```

2. From the resulting text-based menu, select option 5, Set Recipient Mail Address For Email Alert:

```
1  : Show Mail Configuration Information
2  : Disable Email Alert
3  : Enable Email Alert
4  : Send Test Mail
5  : Set Recipient Mail Address for Email Alert
6  : Relay Host IP Configuration
7  : Quit
Select an item  => : (1..7) [7] 5
```

3. From the Mail Config menu, select option 12, FRU Class:

```
            Mail Config  Menu
        ---------------------------
        1  : Environment class
        2  : SFP class
        3  : Port class
        4  : Fabric class
        5  : E-Port class
        6  : F/FL Port (Optical) class
        7  : ALPA Performance Monitor class
        8  : End-to-End Performance Monitor class
        9  : Filter Performance Monitor class
        10 : Security class
        11 : Resource Monitor class
        12 : FRU class
        13 : Quit
Select an item  => : (1..13) [13] 12
```

4. You are now prompted to enter an email address for the alert emails to be sent to. Enter your email address or email group address (I'm using my own here):

   ```
   Mail To: [NONE] nigelpoulton@hotmail.com
   Email Alert configuration succeeded!
   ```

   This message shows that the configuration of an email address for FRU-related alerts has succeeded. However, you have not yet configured alerts to be sent for FRU conditions.

5. From the main menu, select option 3, Enable Email Alert:

   ```
   1  : Show Mail Configuration Information
   2  : Disable Email Alert
   3  : Enable Email Alert
   4  : Send Test Mail
   5  : Set Recipient Mail Address for Email Alert
   6  : Relay Host IP Configuration
   7  : Quit
   Select an item  => : (1..7) [7] 3
   ```

6. From the Mail Enable menu, select option 12, FRU Class:

   ```
               Mail Enable Menu

           ---------------------------
           1  : Environment class
           2  : SFP class
           3  : Port class
           4  : Fabric class
           5  : E-Port class
           6  : F/FL Port (Optical) class
           7  : ALPA Performance Monitor class
           8  : End-to-End Performance Monitor class
           9  : Filter Performance Monitor class
           10 : Security class
           11 : Resource Monitor class
           12 : FRU class
           13 : Quit
   Select an item  => : (1..13) [13] 12
   Email Alert is enabled!
   ```

Email alerting has now been configured to send emails to your email address whenever an event occurs relating to field replaceable units.

Storage Resource Management Applications

It's important at the outset here to point out that when referring to SRM tools in this section, we are not talking about VMware Site Recovery Manager. In this section, we are talking about Storage Resource Management tools.

In large storage environments with lots of technologies from lots of vendors, it's easy to end up with management tool sprawl. For example, imagine the following medium-sized environment:

- 3 EMC VNX arrays
- 2 Dell Compellent arrays
- 4 Brocade FC switches

This environment would normally need to be managed by at least the following management tools:

- EMC VNX GUI or CLI
- Dell Compellent GUI or CLI
- Brocade GUI or CLI

In some cases, you may need multiple instances of these management tools. You might need two Dell Compellent GUI instances to manage each of the two Dell Compellent arrays. This can easily get out of hand in medium and large environments.

To avoid this management tool sprawl, third-party companies and storage vendors have developed tools that attempt to manage heterogeneous storage environments from a single tool. We call these *Storage Resource Management (SRM) tools*.

Under the hood, SRM applications often use Storage Management Initiative Specification (SMI-S) to communicate with and control heterogeneous devices. As a result, they're limited to the capabilities that each vendor's devices expose via SMI-S, and not all vendors have been that good with their SMI-S support. When vendors don't expose functions via SMI-S, SRM tools then have to revert to vendor-specific APIs, narrowing the range of systems supported by the SRM tool and potentially pushing up the cost of the SRM tool.

NOTE We frequently use the term *heterogeneous* in the storage industry to refer to storage technologies from multiple vendors. If you have a storage estate with arrays from multiple vendors, you have a heterogeneous environment. If all your storage equipment is from a single vendor, then you don't.

Although the intention behind SRM tools is good, the reality is that they're a nightmare to implement and maintain.

In the real world, SRM tools tend to have the following requirements and attributes:

- Dedicated database backend
- Powerful servers with lots of CPUs and RAM
- Lag behind native element managers
- Complex licensing
- One or more experts needed to manage them (experts at using the SRM tool)
- Poor performance

As a result, rather than simplifying administration by reducing the number of tools required to manage equipment in the storage estate, most companies end up deploying an SRM tool but still having to keep the native element managers for each platform. And they pay a lot of money for the privilege. The two major reasons behind this are as follows:

- Most storage administrators don't trust SRM tools to do the day-to-day management of their storage arrays and switches.
- SRM tools tend to lag at least six to twelve months behind native element managers when it comes to supporting new features.

Often, SRM tools are used for estate-wide reporting, and native element managers are used for management and troubleshooting of the environment. At the end of the day, a one-size-fits-all tool is never going to be as good and have as deep a knowledge of a technology as that technology's native element manager. For example, the native element manager for an EMC XtremIO is always going to be able to configure and report on an EMC XtremIO array better than a heterogeneous third-party SRM tool.

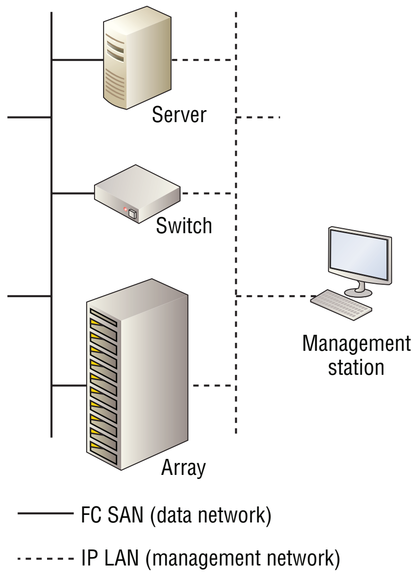
Management Protocols and Interfaces

There are several ways to manage storage devices in your estate. Most of these tend to be network based (IP). Before you explore the most common ways to manage storage devices, let's consider the terms *out-of-band* and *in-band*.

In terms of device management, *out-of-band* management is the most common and requires a dedicated interface for management traffic. A common example in the server world is lights-out management interfaces such as Integrated Lights-Out (iLO) in HP ProLiant servers. Usually, these dedicated management interfaces have their own dedicated processors that allow the system to be managed even when the OS is down or sometimes even when the power isn't turned on. Out-of-band management often has its own dedicated network so that management traffic doesn't interfere with data traffic, as well as providing management access to the device if the main production data network is ever down. A common and simple example in the FC SAN world is to have FC SAN–attached storage devices managed over the IP network rather than over the FC SAN network that is used for data traffic.

Figure 12.9 shows an FC storage array, an FC switch, and a server that are each connected to the FC SAN for data traffic and are also connected to a separate IP network that is dedicated to management traffic.

FIGURE 12.9 Out-of-band management



On the other hand, in *in-band* management, management traffic and data traffic share the same network and potentially the same network interfaces. It can also be the case that in-band management requires the system and the OS to be up and running in order to be able to manage the system.

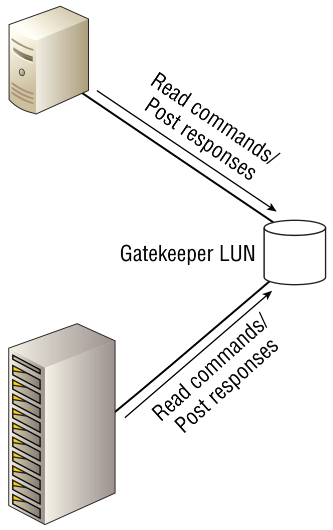
An example of in-band management in FC SAN environments is the use of so-called *gatekeeper devices* or *command devices*. These devices are special LUNs (volumes) that are presented from the array to a management host over the FC SAN. Management hosts can then post commands to these devices, and the array posts responses to the commands. This effectively allows you to manage the array over the FC SAN rather than an IP network. This kind of configuration is usually intended for scripting of replication and snapshot tasks, where the host runs scripts to control the snapshot and replication status of volumes on the array.

Figure 12.10 shows an example of an FC storage array presenting a gatekeeper device over the SAN (in-band) to a host.

NOTE

An easy way to conceptualize in-band and out-of-band management is to think of the network used for data (nonmanagement traffic) as a piece of string, or a band. If management traffic is sent over that same band, then it's in-band. If it's sent over a separate network, it's sent over a separate band and therefore is out-of-band.

FIGURE 12.10 In-band FC management



Command-Line Interface

The *command-line interface (CLI)* has always been synonymous with hard-core management, and it's often been the case that advanced configuration and advanced management tasks could be performed only via the CLI. For this reason, sometimes the CLI was touched only by advanced or expert users. The phrase *real men don't click* is occasionally used to suggest that experts don't waste their time with GUIs and their point-and-*click* interfaces.

When it comes to estate management via the command line, two main options are available:

- Telnet
- Secure Shell (SSH)

They're both TCP/IP-based network protocols that are used to provide command-line access to remote systems. However, SSH is the most popular choice these days, and for good reason. Telnet is a security nightmare. It doesn't support authentication or encryption, and it sends passwords over the network in cleartext! Don't use it. Use SSH instead, and if possible use SSH-2, because it's more secure.

NOTE Although Telnet and SSH work over the network, it is also possible to get a command-line interface to some systems by plugging in a serial cable—sometimes called a console cable. To do this, you usually need to have your laptop with you and be in front of the system. One end of the serial cable (console cable) plugs into a serial port on your laptop, and the other end plugs into the serial port on the system you want to manage. However, not many laptops these days come with serial ports. In these situations, you can either use a USB-to-serial converter/cable, or connect a device that does have a serial port and then access that device over the network and remotely use its serial port. These console connections are often required to perform initial setup tasks such as configuring management IP addresses and the like.

SSH supports mutual authentication, via certificates, as well as encryption, making it massively more secure than Telnet. It's a client-server-based protocol with storage arrays and SAN switches needing to run an SSH server daemon in order for you to connect to and manage them with an SSH client. Most storage arrays and SAN switches run SSH servers and are therefore natively manageable via SSH.

SSH is a TCP-based protocol and operates, by default, over TCP port 22. However, it is possible to administratively change this to any other TCP port you wish.

NOTE Technically speaking, SSH is a suite of three secure tools/utilities: slogin, ssh, and scp. Each of these are secure versions of the popular Unix utilities rlogin (remote login), rsh (remote shell), and rcp (remote copy). The versions in SSH each start with an *s* to indicate they are *secure* versions: secure login (slogin), secure shell (ssh), and secure copy (scp).

One area in which the CLI is far superior to graphical interfaces is when it comes to automation of tasks. It's common for storage administrators to schedule tasks that run via the CLI and scripts—collections of commands that execute one after another.

Graphical User Interface

Most storage devices support management via a *graphical user interface (GUI)* for performing basic day-to-day tasks. Generally speaking, the GUI is less powerful than the command line, usually for the following reasons:

- Tasks can't be automated through the GUI.
- GUIs often provide access to only the most common commands and functions.

Many modern GUIs are web based and allow you to connect to them via a standard web browser. In these cases, you should always configure them to accept connections only over HTTPS instead of plain HTTP. HTTPS can be thought of as a secure version of HTTP, though in reality it's just plain old HTTP running on top of the secure SSL/TLS protocol. As long as your storage device supports HTTPS, you'll be able to connect to it via HTTPS from any supported web browser.

An HTTPS connection is initiated by adding an *s* to the end of the *http* portion of a URL, as shown here:

```
http://technicaldeepdive.com
https://technicaldeepdive.com
```

SMI-S

As noted earlier, SMI-S is an acronym for *Storage Management Initiative Specification*. It's an SNIA-based initiative that aims to provide a common web-based standard for managing heterogeneous storage devices. Instead of each and every storage device having its own proprietary management interface and management tools, you have them all implement SMI-S and be manageable from any SMI-S-based management software. A potential end goal is to have a single pane of glass to manage a large heterogeneous storage environment—such as an SRM application. That is a state of nirvana if there ever was one!

Sadly, this nirvana has never been realized. Precious few, if any, storage admins trust the day-to-day management of their storage devices to anything other than the vendor-specific tools (native element managers) that come with the devices. Also, vendor support of SMI-S has been patchy at best, with new features often not being added to SMI-S, or at least lagging a long way behind.

However, SMI-S is useful in centralized reporting and monitoring of heterogeneous storage environments. It's common for companies to use a single enterprise reporting and monitoring tool for their entire storage estate, even if that estate has equipment from multiple vendors.

Real World Scenario

Example of SMI-S Shortcomings

One company that was an early implementer of thin provisioning from one of the major storage vendors found itself in an awkward position when they realized that their third-party SRM tool that they were using to report on the estate didn't support or understand thinly provisioned volumes. All of their monthly capacity reports were of no use on their arrays that they were implementing thin provisioning on. Despite pressuring the SRM vendor, it was about nine months before the vendor shipped a version of the product that supported thinly provisioned volumes.

For the record, SMI-S is implemented as a client-server model. In this model, the servers are the storage devices such as arrays, switches, and HBAs. The SMI-S client is the management tool.

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is a popular IP-based protocol for managing and monitoring devices on an IP network. More often than not, though, SNMP is used for monitoring the health and status of network-attached devices, rather than managing and making configuration changes to them.

SNMP is client-server based:

- Clients are the devices, such as arrays, switches, and HBAs, that are managed and monitored. Sometimes clients are referred to as *nodes* or *managed devices*.
- Servers are the applications that monitor and manage SNMP clients. For example, SNMP servers commonly receive alerts from clients (referred to as *traps*) when components fail. Also, in a read/write SNMP configuration, servers push configuration changes to SNMP clients. Sometimes we call servers *network management systems (NMSs)*.

SNMP is an application-layer, IP-based protocol that talks over UDP ports 161 and 162, although secure versions that operate over TLS talk on ports 10161 and 10162. Therefore, all communications between SNMP clients and servers occur over IP.

NOTE

From a security perspective, the most recent version of SNMP—version 3 (SNMPv3)—is the best choice. All communication, including passwords, were sent in cleartext with SNMPv1 and SNMPv2. In contrast, SNMPv3 supports all three of the major network security features: authentication, integrity, and privacy (encryption). In practice, SNMPv3 is gaining in popularity, and it's increasingly common for SNMP-capable systems to support it.

Exactly what can be managed and monitored on a particular SNMP-capable device (client) depends entirely on the device itself, and as a result, some devices offer extensive SNMP capabilities, whereas other don't. SNMP-capable devices maintain a management information base (MIB) that defines what settings can be monitored and managed.

Summary

In this chapter we started out by talking about the importance of capacity management and capacity reporting, and then talked briefly about some of the major technologies that can influence capacity management. We then moved on to performance management, and stressed the importance of performance baselines, and then discussed the major performance metrics used in the storage industry. We then talked about common factors that can influence storage performance and mentioned some of the tools used to measure performance. We finished the chapter by discussing management protocols such as SMI-S and SNMP, as well as graphical user interfaces and command-line interfaces.

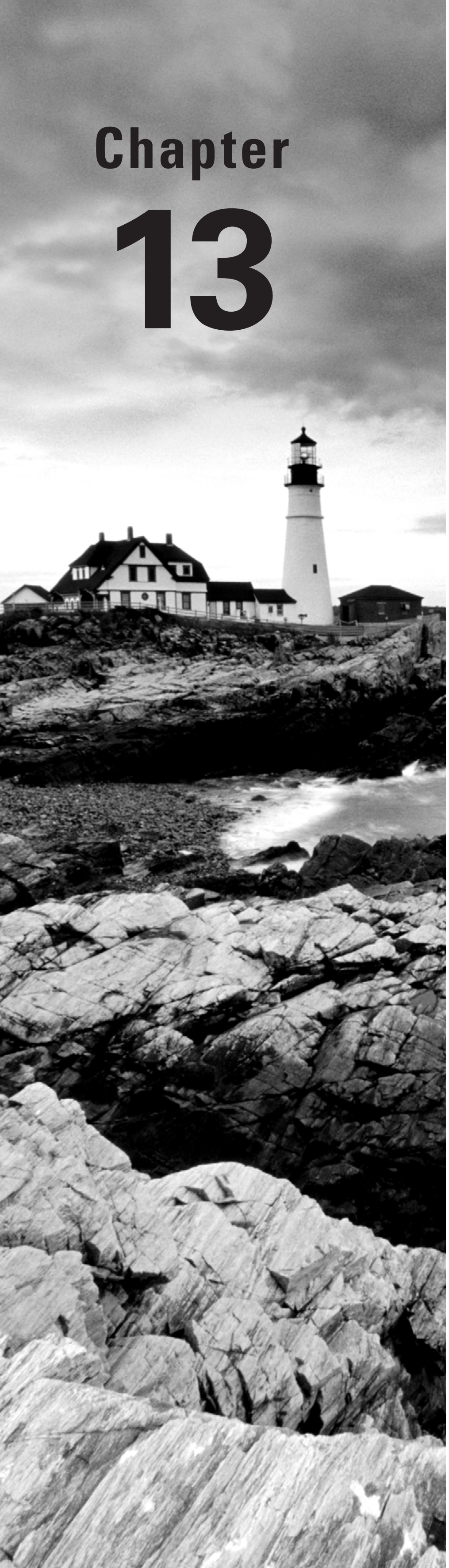
Chapter Essentials

Capacity Management First and foremost, capacity management is about making sure your storage estate doesn't run out of capacity. Once you're on top of that, it's then about making optimal use of the capacity you have. This includes things like making sure data that needs to be accessed quickly is on fast media. Reporting is also a major component of capacity management—knowing who's using what, and what growth projections look like.

Performance Management Performance management is about making sure the end-to-end performance of the storage estate is adequate. Poor performance can slow down applications to the point that they are useless. An important part of performance management is baselining, which enables you to compare today's performance with how it *should* be.

Alerting Alerting is a means of ensuring that you and other people are made aware of important conditions and events in the storage estate. These conditions and events include things like failed components and changes in the state of services, as well as capacity- and performance-related events.

Chapter 13



The Wider Data Center Neighborhood

TOPICS COVERED IN THIS CHAPTER:

- ✓ **The importance of the data center**
- ✓ **Data center tiers**
- ✓ **Racks, cabinets, and rows**
- ✓ **Power and cooling**
- ✓ **Structured cabling**
- ✓ **Access control**
- ✓ **Fire prevention**

This chapter covers the central role and importance of the data center, as well as the high-level principles of data center design. You'll learn about the tiers of the data center, as defined by the Uptime Institute. You'll also learn about the major components of the data center, including raised floors, racks, power systems, cooling systems, and cabling. This chapter presents in some detail the major components of a structured cabling scheme and outlines best practices that will make day-to-day management of the data center as simple as possible, while still allowing the data center to be flexible enough to cater to tomorrow's requirements. Finally, the chapter concludes by listing some good practices for working in a data center.

Data Center Design

The first thing to know about modern data centers is that they are vital to any organization that values its data. If you value your data, value your data center! If you don't value your data center, you don't value your data!

The data center is more than just the sum of its component parts. It is rightfully a system in and of itself. Gone are the days of a data center being a grubby spare room used to hide away IT equipment. In the modern world, instead of being merely a *technology room,* the data center is more of a *technology engine*—an integrated system of computing, storage, network, and power and cooling, with its own levels of performance, resiliency, and availability. As such, the data center requires a proper design, as well as appropriate levels of management and maintenance. Again, if you care about your data, you need to care about your data centers.

In many ways, the data center is the perfect home for computing, network, and storage devices—a habitat built from the ground up to keep them all in perfect working condition. Data centers provide not only physical security for IT equipment, but also cooling, humidification, grounding, and power feeds that are all highly optimized for IT equipment. As a human being, you won't want to spend too much time in a tier 3 or tier 4 data center. If you do, you'll end up with sore, dry eyes and probably irritated ears. You'll also probably find the varying temperatures uncomfortable too, as some areas of the data center will be too cold for comfort, whereas others will be too warm. Data centers are concerned with keeping IT equipment, not human beings, in working order.

With all of this in mind, let's take a quick look at the *tiers* of the data center that we commonly refer to when talking about data center uptime and availability.

Data Center Tiers

The most widely accepted definition of data center tiers is the one from the Uptime Institute (`http://uptimeinstitute.com`). The Uptime Institute is an independent division of the 451 Group, dedicated to data center research, certification, and education. These tiers are also borrowed and referenced by the Telecommunications Institute Association (TIA), making them pretty much the de facto standard.

The reason for classifying data centers into tiers is to attempt to standardize how we define tiers and talk about data center uptime and availability.

NOTE The Telecommunications Industry Association (TIA) introduced an important standard in 2005 relating to cabling within a data center. This TIA-942 standard has been instrumental in standardizing and raising the level of cabling within the data center. TIA-942 includes an annex that references the Uptime Institute's data center tier model.

At a high level, the Uptime Institute defines four tiers of a data center:

- Tier 4
- Tier 3
- Tier 2
- Tier 1

As indicated by this list, tier 4 is the highest tier, and tier 1 is the lowest tier. This means that tier 4 data centers have the highest availability, highest resiliency, and highest performance, whereas tier 1 data centers have the lowest.

On one hand, a tier 4 data center will have redundant everything—power, cooling, network connectivity, the whole works. On the other hand, a tier 1 data center could be little more than your garage with a single power circuit, a mobile air-conditioning unit, and a broadband connection.

NOTE Not all data centers are audited and certified by the Uptime Institute. However, data centers can still be built and maintained to standards that are mapped to formal Uptime Institute tiers. To formally refer to a data center as, for example, tier 3 or tier 4, the data center should be formally certified.

Let's talk about some of the high-level features and requirements of the data center tiers specified by the Uptime Institute.

Tier 1

Tier 1 is the most basic, and lowest availability, data center. It is kind of like your garage at home. There's no requirement for things such as multiple power feeds or an on-site diesel generator. There's also no requirement for multiple diverse network connections. Basically, there is no requirement for any level of component redundancy. You could almost build one yourself at home! As a result, tier 1 data centers are extremely susceptible to unplanned downtime, so you almost certainly don't want to run your business from one.

Tier 2

Tier 2 takes things up a considerable level from tier 1. Tier 2 data centers have redundant on-site generators and uninterruptable power supplies, as well as some component-level redundancy. However, tier 2 data centers still require downtime for certain planned maintenance activities, such as annual power maintenance and testing.

Tier 3

Tier 3 data centers take the foundation of tier 2, and add N+1 redundant power and cooling paths (for example, all equipment must have multiple power supplies and power feeds). These redundant paths need to work in only active/passive mode, although each path must be capable of handling the entire load on its own. This active/passive mode of operation allows you to perform planned maintenance without incurring downtime, but does not necessarily protect you in the event of an unplanned issue on active power paths. Tier 3 data centers require on-site generators that are capable of powering the entire data center for extended periods of time.

Tier 4

A tier 4 data center is the highest tier currently defined. A tier 4 data center can ride out *unplanned* outages of essential services such as power, cooling, and networking without incurring downtime. All of these major systems are implemented as *multiple N+1 redundant* and physically isolated, which means the power systems might have two independent paths, each of which is N+1 redundant. Tier 4 data centers also require on-site generators that are capable of powering the entire data center for extended periods of time.

If you can afford it, you probably want to run your business out of a tier 4 data center. If you're partnering with a private cloud supplier, you want to ensure their data centers are tier 3 or tier 4 data centers.

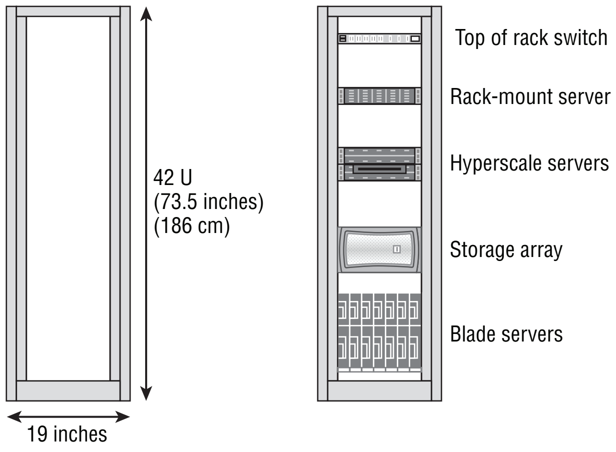
Data Center Overview

Let's now take a look at some of the major components of the data center.

Racks

Almost all IT equipment in data centers is housed in standardized racks referred to as *19-inch racks*. Figure 13.1 shows the front view of two side-by-side industry-standard 19-inch racks. One rack is empty, and the other rack is partially filled with various computing, storage, and network hardware.

FIGURE 13.1 19-inch racks



These 19-inch racks get their name from the fact that, when viewed from either the front or the back, they are about 19 inches wide. This standard width, coupled with standardized depths, allows for IT equipment such as servers, storage arrays, and network switches to come in standard sizes and fit almost all data center racks. This makes installing equipment in the data center so much easier than it was before manufacturers had standardized on the 19-inch rack.

NOTE

According to the EIA-310D standard, 19 inches is the width of the mounting flange on the equipment. The space between the rails is 17.72 inches (450 mm), and center-hole spacing is 465 mm.

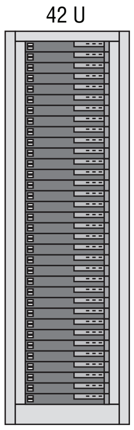
When talking about the *height* of 19-inch racks, we usually talk in *rack units (RUs)*, with a standard RU being 1.75 inches. More often than not, we shorten the term *rack unit* to simply *unit* or *U*. Hence the popular term, and rack size, of *42 U*, meaning 42 rack units in height, or 73.5 inches. We also measure the height of servers, switches, and storage arrays in terms of RU, or U. For example, a single 42 U rack can house up to 24 2-U servers stacked directly on top of each other, as shown in Figure 13.2. It should be noted, however, that although a single rack can physically be fully populated, the data center may not be able to supply enough power and cooling to allow you to fully populate the rack.

The process of installing equipment in a 19-inch rack is referred to as *racking*, and the process of removing the equipment is known as *unracking*.

WARNING

Make sure that when you *rack* and *unrack* equipment, you use proper lifting techniques. More often than not, there should be two people involved, so you don't injure yourself or cause damage to equipment.

FIGURE 13.2 19-inch rack filled with 24 2-U servers



WARNING

Heavy equipment should be racked in the bottom of the rack so it doesn't topple over when leaned against or when sliding heavy equipment in and out of the rack. Racks should be bolted together or have special stabilizing feet attached to provide additional stability and resistance to toppling.

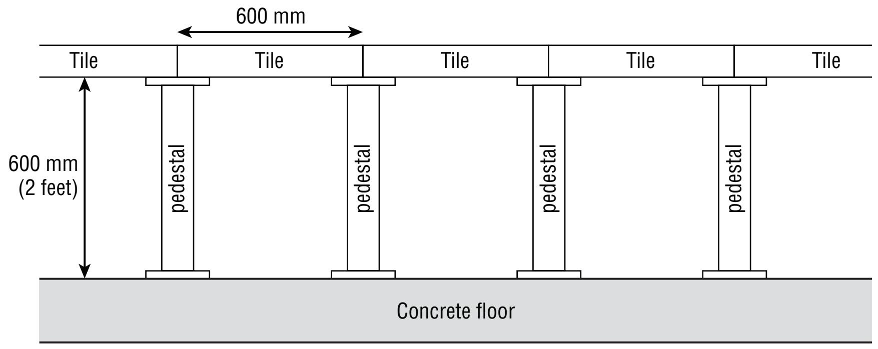
Sadly, storage arrays are one of the worst offenders when it comes to requiring nonstandard specialized racks. It's not uncommon for data centers to have entire rows dedicated to storage equipment, because storage arrays often come in custom racks and can't be installed in standard data center racks. Fortunately, though, most storage vendors are starting to support their equipment when installed in industry-standard 19-inch racks.

It is required by U.S. electrical code to ground racks, and sometimes equipment within racks, to protect people and equipment from electric shock and static. For this same reason, it is highly recommended to wear antistatic wristbands when working with equipment in the data center, especially if taking the lid off equipment.

Flooring

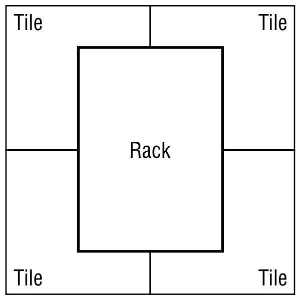
It is common practice in modern data centers to have raised floors. A raised floor is an artificial floor, usually made of 600 mm × 600 mm floor tiles that rest on top of pedestals. The floor tiles are removable, allowing engineers to lift them and gain access to the void between the raised floor and the solid floor beneath. This floor void is commonly used to route network cables and power cables, as well as potentially channel cold air. Figure 13.3 shows an artificially raised floor sitting 2 feet above a concrete floor on supporting pedestals.

FIGURE 13.3 Raised floor and void



It is important to know how much weight the raised floor can sustain, so you don't load a rack with more weight than the floor is rated to hold. You should always consult with the experts when it comes to the load-bearing capacity of your raised data center floor. It is common practice to install racks so they span multiple floor tiles. Quite often, each floor tile is capable of taking a certain weight, and standing a rack across four tiles, as shown in Figure 13.4, can sometimes give you the strength and weight-bearing capacity of all four tiles. However, make sure that you seek the advice of a knowledgeable expert before making this assumption.

FIGURE 13.4 Aerial view of data center rack spread over four floor tiles



Rows

In most data centers, racks are organized into rows, as shown in the aerial view provided in Figure 13.5.

In this manner, racks can easily be located by giving their row location as well as their rack location. For example, rack 3 in row A is highlighted in Figure 13.6.

FIGURE 13.5 Racks organized in rows

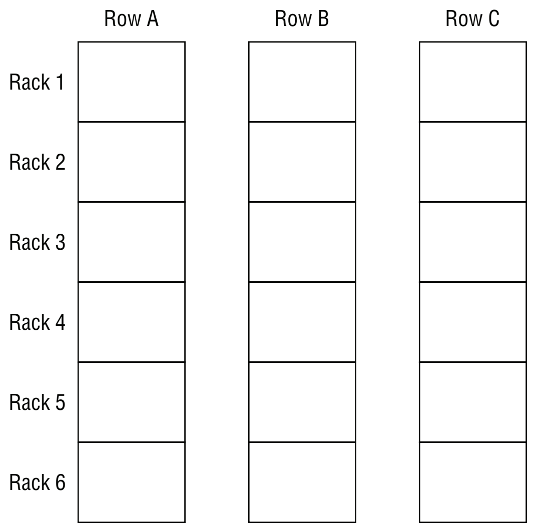
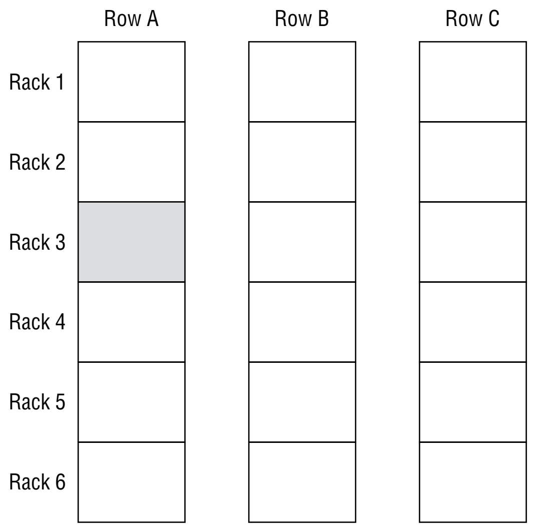


FIGURE 13.6 Locating rack 3 in row A



Later in the chapter, you will see how rows are important when it comes to power and cooling. For now, it's enough to know that data center design and management are far simpler if you line up racks in neat rows.

Power

Power is one of the most important considerations for modern data centers! If you thought the world's insatiable appetite for more and more storage was off the scale, then demand for power in the data center is right there with it!

Data centers need power and often can't get enough of it. In fact, it's not uncommon for some of the larger data centers to draw more power than an average-sized city. But it's not just the servers, storage, and networking equipment that suck the power. Heating, ventilation, and air conditioning (HVAC) systems consume a lot of power. In some places in the world, HVAC consumes more power than the IT equipment, meaning that many data centers consume more power keeping the temperature right than they do keeping equipment powered on.

But it's not just the overall data center as a whole that is struggling with power. Many data centers are struggling to provide enough power and cooling to individual racks. For example, modern servers draw more power and pump out more heat than ever before, and data centers that can provide only in the region of 5–6 kW to each rack just aren't supplying enough power to be able to fill the rack with servers. If your data center can't provide enough power for cooling to enable you to fill your data center racks, your racks may have to operate up to half empty. Many modern data centers now provide 20 kW or more to each rack, so every inch of rack space can be used!

NOTE In 2011, Google announced that its data centers in 2010 were continuously consuming 260 megawatts (MW) of power. That's 260,000,000 watts, which equates to more power than that used by Salt Lake City, Utah. Of course, Google is not the only company to have large data centers that consume a lot of power, and Google also claims to do a lot to offset and minimize its impact on the environment.

Because of the power constraints we currently operate under, vendors are placing significant focus on improving the efficiency of power supply units (PSUs) in the back of servers, storage, and networking equipment. For example, if it costs you $500,000 a year to power your IT equipment, a 20 percent improvement in the efficiency of PSUs could significantly reduce your annual electricity bill.

The industry uses the power usage effectiveness (PUE) rating to measure efficiency of a data center. PUE is determined by the following formula:

$$x = \frac{\textit{Total power supplied to the data center}}{\textit{Total power consumed by IT equipment}}$$

Where x is the efficiency rating. A PUE of 2 means that you have a single watt of overhead for every watt you supply to your actual IT equipment (server, storage, network). A PUE of 4 would mean you have 3 watts of overhead for every watt used to power your equipment. So, obviously, a lower PUE rating is better.

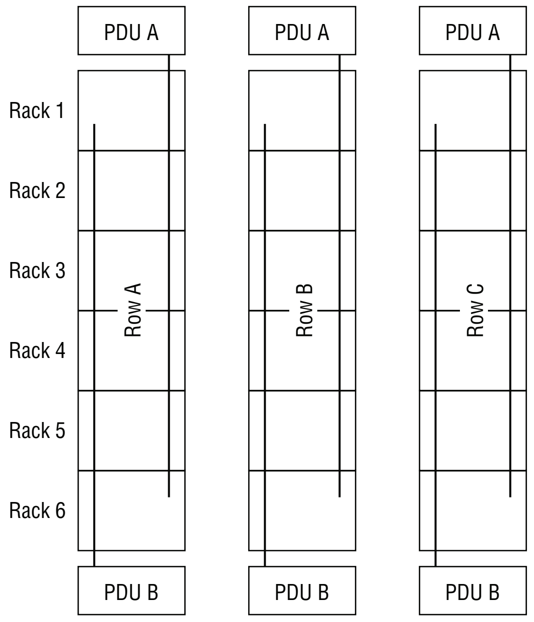
Quite often, data centers have multiple independent power feeds (ideally from separate utility companies if that's possible). This allows them to lose power from one utility company and not have to run on power from the on-site generator.

NOTE

To provide power to the data center when utility power is lost, most data centers have in-place backup power systems based on diesel-powered motors. If utility power is lost, these on-site diesel generators kick in and power the data center until utility power is returned. It is common practice to maintain enough fuel on-site to be able to power the data center at full capacity for a minimum of 48 hours. It's obviously important to perform regular (at least annual) tests of these on-site generators so you know they can be relied on when called into action. As an example of the importance of such generators, when Hurricane Sandy hit in 2012, many data centers in the New York and Newark areas had to rely on generator power for extended periods of time.

Within the data center itself, it's common to provide two independent power feeds to each and every rack. This power is often supplied via independent power distribution units (PDUs) at the end of each row, as shown in Figure 13.7.

FIGURE 13.7 Power distribution to data center rows

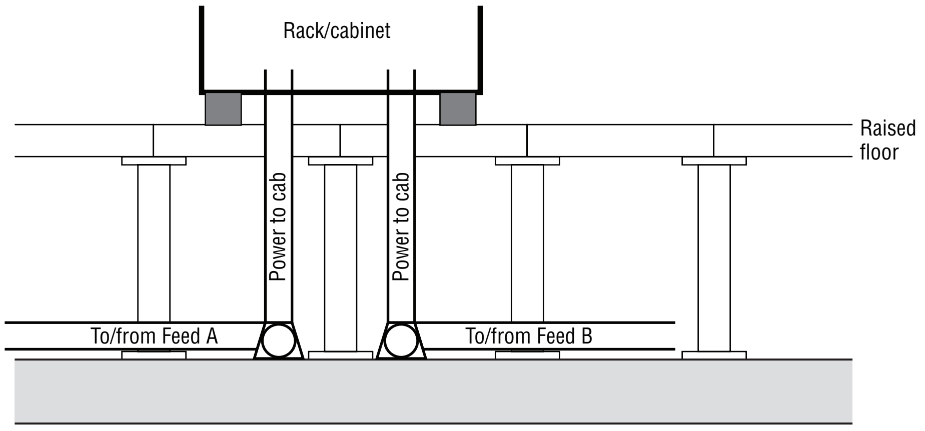


In Figure 13.7, each row is fed by a power feed from PDU A as well as a feed from PDU B. For each row, PDU A is independent of PDU B, meaning that one PDU can fail, or the power supplied to it can fail, without the other being affected. For example, if PDU B fails for row C, PDU A will not be affected and will continue to provide power to equipment racked in row C. This redundant power configuration allows for computing, storage, and networking equipment to be dual-power fed and survive many power failure scenarios. This also allows you to perform maintenance on one power feed without having to power equipment down or run on power from the on-site generator.

It is also common for all power to be protected by an uninterruptable power supply (UPS) that ensures that power is clean and spike free. UPS systems often provide a few seconds or minutes of power in the event that utility power is briefly interrupted. This means that computing, storage, and networking equipment is unaffected by brief spikes or drops in power.

Power is usually fed to the racks via the floor void, as shown in Figure 13.8.

FIGURE 13.8 Power to cabinets via floor void



Power to each rack usually has its own circuit breaker on the PDU at the end of the row, allowing you to isolate power on a rack-by-rack basis.

In data center facilities that do not have raised floors, it is also possible to route power cables overhead, and have industrial and multiphase power plugs and sockets suspended from the ceiling so you can easily connect them to rack-based power strips. As with routing cables in the floor void, routing them overhead keeps them out of the way when performing day-to-day work in the data center.

NOTE Power is often referred to as one of the *mechanical and electrical (M&E) subsystems* of the data center.

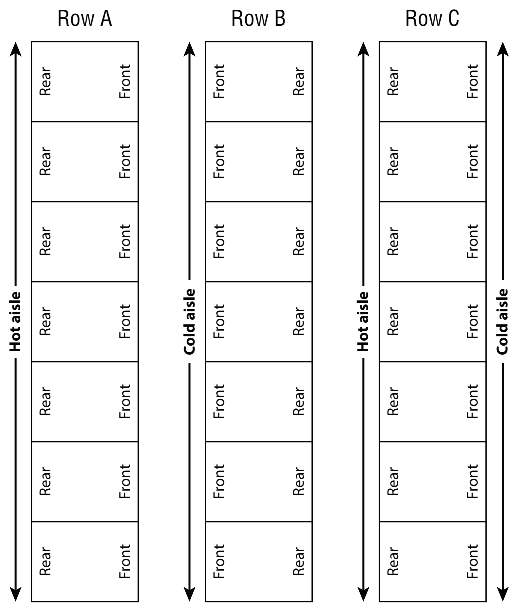
Cooling

Cooling is big business in data centers, with a lot of power being expended on keeping the data center cool. To reduce the amount of energy required to keep equipment cool, most data centers operate on a form of the *hot/cold aisle principle*. Most modern IT equipment is built so that fans suck in cold air from the front and expel hot air from the rear. In a hot/cold aisle configuration, racks are installed so that all racks in a row face the same direction, and opposing rows face each other in a back-to-back configuration. This means that equipment in adjacent rows is always facing either back-to-back or front-to-front, as shown in Figure 13.9.

NOTE

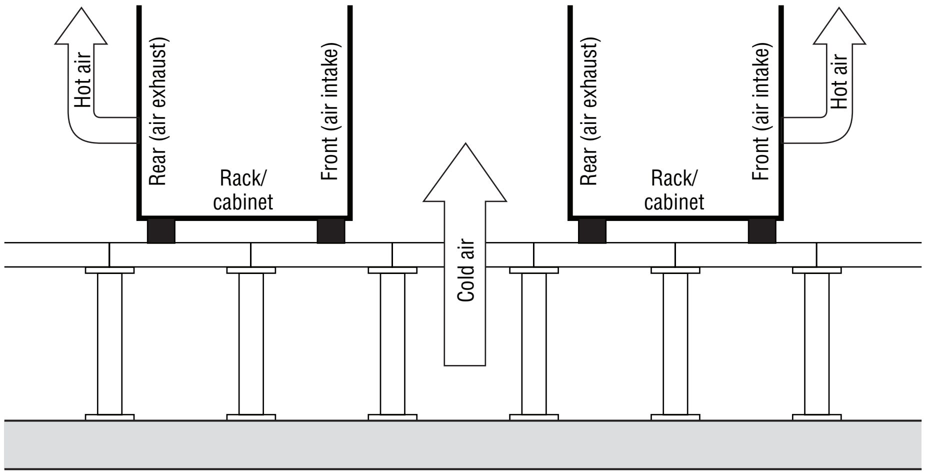
As much as cooling is a function of power, it is also a function of equipment placement. One Uptime Institute report suggests that equipment higher in the rack is far more likely to fail than equipment lower in the rack. This is because cool air is warmed as it rises, making the temperature at the top of the rack higher than at the bottom.

FIGURE 13.9 Hot/cold aisle configuration



The kind of hot/cold aisle configuration shown in Figure 13.9 helps keep hot and cold air separated. This makes for a far more efficient data center cooling system than if the hot air that was blown out of some equipment was then sucked into the front of other equipment; trying to keep equipment cool by using hot air isn't a good idea. However, even with hot/cold aisle data centers, hot and cold air still manages to mix, and as a result, some data centers decide to *enclose* either the hot aisle or the cold aisle. For example, in a data center where cold air is kept within an enclosed aisle, hot air from outside cannot get into the enclosed area and raise the temperature. It is also popular to pump cold air around the data center through the floor void, and allow it up into the cold aisle via perforated floor tiles, as shown in Figure 13.10.

FIGURE 13.10 Cold air routed through the floor void



Most modern data center racks come with perforated front and rear doors so racks can breathe; air can come in and out of the rack through the tiny perforated holes in the front and rear doors. If your racks don't have perforated front and rear doors, you should seriously consider removing the doors entirely. In fact, removing the front and rear doors is not an uncommon practice in large data centers.

NOTE Some equipment, especially older network equipment, doesn't conform to front-to-rear airflow. Some equipment sucks in cold air from one side and blows hot air out the other side, and some equipment even takes cold air in from below and expels hot air from the top. If you operate a data center that is built around a hot/cold aisle design, you should try to avoid this kind of nonstandard equipment in the same areas of the data center.

It's not uncommon for data centers to be located in strategic geographic areas that provide opportunities to leverage local environmental conditions in order to make the data center as green and environmentally friendly as possible. Examples include areas with cool air that can be channeled into the data center and used to cool equipment or areas with long sunny days that enable good use of solar power.

Amid all this discussion about racks, rows, power, and airflow, it's easy to forget that the purpose of all this is to make the best possible environment for servers, storage, and networking equipment to operate in, but also to do so at the most efficient cost.

Server Installation in a Data Center Rack

Now that we've talked a lot about racks, power cooling, and related matters, Exercise 13.1 walks you through the common steps required to install a new server into a data center rack.

EXERCISE 13.1

Racking a New 2U Server

This exercise outlines some of the steps required to install new equipment in a data center rack.

1. Determine whether the rack where you are installing your new equipment has the capacity for that equipment. Check for the following:
   - Is there enough physical space in the rack? The rack will need at least two free rack units.
   - Are there enough free sockets in the power strips in the rack to take the power cables from your new server?
   - Is there enough spare power to the rack to handle the power draw of your new server?
   - Is there enough cooling to the rack to handle your new server?
   - Are there appropriate copper and fiber cables to the rack?
   - Will you have to remove floor tiles for any of the network or power cabling work?
   - Can the floor and rack take the weight of adding your new 2U server?
2. After you have determined that there is enough space, power, cooling and so on, you need to plan the installation of your new server. This planning often includes many of the following considerations:
   - When, and by whom, will your new server be delivered? Most data centers do not allow delivery of equipment without prior notification. You may well need to pre-inform the security office at the data center and provide them a description of the equipment as well as the name of the company delivering the equipment.
   - Who will rack your equipment? You will need two people to physically rack the equipment.

- When can the equipment be racked? If it will be racked into a rack in the production zone of your data center, the physical racking work may need to be performed on a weekend and may require an authorized change number to be formally agreed upon by a change management board.
- Do you have rail equipment to allow your server to be physically racked?
- Are you required to use anti-static wrist-bands, anti-static mats, or other specific equipment? If so, where are they stored?
- Are there guidelines or protocols that you are required to follow? Be sure to review them in advance, and follow up with the appropriate parties with any questions you have.
- What paperwork or forms do you need to fill out in advance, and how far in advance do you need to fill them out?
- Do you need to engage with the network and storage teams so that your new server is physically and logically configured on the IP and storage networks?

3. After you have everything planned, the day arrives for you to install your new equipment. Before heading down to the data center, ensure that all your electronic paperwork is in order. Make sure that the security staff at the data center are expecting you and that they will allow you on site. Also make sure your equipment has been delivered and find out where it's currently being stored.
4. When you arrive and have gained access to the site and the computer room with your new server, follow a procedure similar to the following:
   a. Locate the rack where your server will be racked.
   b. Open the rack and ensure that there are no obstructions to installing your new server. Be careful with power and network cables to existing servers and storage. You definitely don't want to knock any of them out.
   c. Unpack your server. It is standard practice at most data centers for you to unbox new equipment in a build room and *not* in the computer rooms of the data center.
   d. Install the racking equipment (rails). You may be required to use an antistatic wristband or antistatic mat while doing the work.
   e. Lift your server onto the rails and gently slide it into place, ensuring you don't unplug or damage any other cables while doing so.
   f. Cable up the network and power, including removing any required floor tiles.
   g. Power up your server.
   h. If you had to lift any floor tiles, put them back in place.
   i. Ensure that all other systems in the rack look fine, and close the rack doors.

j. Test the server, and follow whatever protocols are required to help the network and storage teams configure your server on the IP and storage networks.

k. Dispose of any packaging that your equipment was delivered in.

That is all there is to it. Of course, every scenario is unique. As you perform these steps, you will probably find that your process is different at several points along the way, but this exercise is a good blueprint for the general steps you follow.

Everything we've discussed so far is about creating an optimal environment for business applications and IT equipment to operate in. Now let's take a look at data center cabling.

Data Center Cabling

If you didn't already know how vital power and HVAC are to the data center, hopefully you do now that you have read the previous sections. Let's move on and discover how equally vital good cabling is.

Following Standard Principles

Good, well-thought-out, and properly installed cabling is an absolute must for any data center. It will make the day-to-day management and the future growth of the data center so much simpler than they would otherwise be. There are several overarching principles to account for when cabling a data center. They include the following:

- Planning for the future
- Testing
- Documenting

When planning for the future, try to use cables that will last for 10 years. This includes the robustness of the cable as well as its capability to transmit the kind of data rates (bandwidth) that are anticipated for the next 10 years. As an example, if you're installing copper twisted-pair cabling, make sure to install the latest *category* so that it has a fighting chance of being able to transmit 40G Ethernet. The same goes for fiber cables; install the latest optical mode (OM) available, such as laser-optimized OM3 or OM4, which are both capable of 40G and 100G Ethernet (including FCoE). Cutting corners and cutting costs by installing cheap cables might save you some cash today, but in a few years' time, it will come back to bite you. There aren't many more daunting and costly tasks in data center maintenance than re-cabling.

Planning for the future is also more than just choosing the right kind of cable to use. It involves things like having a large-enough floor void and having enough space for conduits between different floors in the building to allow you to work with cabling today (so you can easily stay within recommended bend radius and so on) as well as add cable capacity in the future.

Real World Scenario

Cabling Consumes Space

One company didn't think far enough ahead when installing a computer room at its corporate HQ. The company didn't leave enough space to run cables between floors and ended up having to decommission an entire elevator shaft and reuse it as a cabling duct between floors.

Testing is vital with cabling, especially structured cabling. If you'll be doing the testing yourself, do yourself and your organization a favor, and use high-quality cabling and high-grade testing equipment. If a third party will be doing the cable testing for you, make sure you accept only the highest scores as a pass score. You absolutely do not want to be responsible for structured cabling that was done on the cheap and not tested properly. Faulty cables in a structured cabling system cannot usually be replaced; you just have to live with the unusable cable and make sure you don't connect anything through it. For this reason, make sure you apply the carpenter's motto of *measure twice, cut once*.

TIP

Copper cabling is susceptible to interference from nearby devices, technically known as *alien cross-talk (AXT)*. This can be avoided by understanding and planning so that you can implement cabling best practices. Some cabling best practices include things like not mixing CAT 6A cables running 10G Ethernet in the same ducts and pathways as CAT 5, CAT 5e, and CAT 6. Patch-panel densities can also lead to AXT in copper patch panels.

Documenting cabling is also vital. This includes things easily overlooked such as using labels on patch panels, as well as placing labels on the ends of cables. Cabling is something that will get out of control and become an unmanageable mess if you don't document it.

Using Structured Cabling

The most common, and recommended, form of data center cabling is structured cabling. The term *structured cabling* refers to the fact that it follows a well-defined set of standards. These standards have been established over the years to improve the quality and manageability of cabling within the data center.

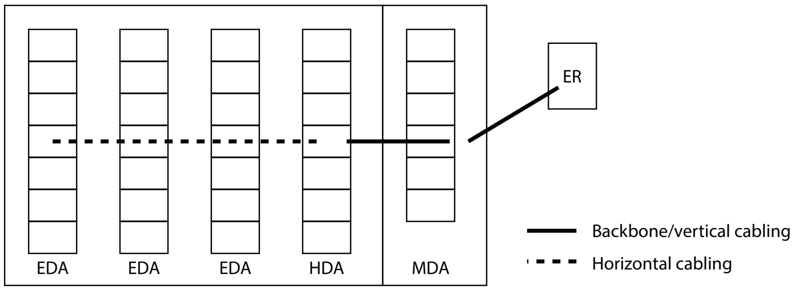
NOTE The opposite of structured cabling is either point-to-point cabling or simple ad hoc cabling, neither of which scale. Point-to-point consists of running cables directly from servers to switches to storage, with the cables usually being routed within and between racks without the use of patch panels. These solutions can work in some small data centers, more often referred to as *computer rooms* or *equipment rooms*, where there is no space or requirement for a serious, long-term, scalable cabling solution.

The de facto standard when it comes to data center cabling is TIA-942, produced by the TIA in 2005. As well as recommending things like using the highest-capacity cabling possible (future-proofing), this standard outlines the following major components of a data center that relate to cabling:

- Entrance room (ER)
- Main distribution area (MDA)
- Horizontal distribution area (HDA)
- Equipment distribution area (EDA)
- Zone distribution area (ZDA)
- Vertical cabling/backbone cabling
- Horizontal cabling

At a very high level, they work together as follows. All external network links arrive into the data center at the entrance room (ER). From here, backbone cabling connects equipment in the ER to the main distribution area (MDA) where your network core resides. More backbone cabling then connects the equipment in the MDA to the horizontal distribution area (HDA), which is where your distribution layer switches—and often core SAN switches—reside. From the HDA, horizontal cabling then runs to the equipment distribution areas (EDAs), where servers, storage arrays, and edge switches reside. This is all shown in Figure 13.11.

FIGURE 13.11 High-level structured cabling



But that's just a high-level picture. Let's look a bit closer at each of these components.

Entrance Room

According to TIA-942, each data center has to have at least one ER, but can have more if required. The ER is where all of your external network connections come in—connections from network providers as well as all connections to other campus networks. Basically, any network that wants to come into the data center *must* come in via the ER. The ER is effectively the buffer between the inner sanctum of the data center and the outside world. In network lingo, we call this the *demarcation point*—where you draw the line between network equipment from your network providers and your own network equipment. Within the ER, you connect your equipment to the equipment from your network providers, and your network providers should have no network circuits or equipment beyond the ER. Anything beyond the ER should be owned and managed by you.

Having multiple ERs can provide higher availability for external links. For example, if all your external links come into a single ER, and there is a power failure, flood, or other incident in that room, you risk losing all external connections to the data center.

The ER can be inside or outside your computer rooms.

NOTE *Computer rooms* are where your MDA, HDA, and EDA (including servers, storage, and network equipment) reside.

Main Distribution Area

The main distribution area (MDA) is inside the computer room and typically houses the core network switches and routers. These are *your* network switches and routers, owned and managed by *you*.

The outward-facing side of MDA is connected to the ER, and the inward-facing side is connected to the HDA. Both of these connections are via *backbone cabling*, sometimes called *vertical cabling*. Backbone cabling is often fiber cabling.

Strictly speaking, the internal side of the MDA can connect directly to your EDAs, or it can connect to the EDAs via HDAs. Which of the two designs you choose will be based on scale. Smaller data centers often omit the HDA and cable directly from the MDA to the EDAs. However, larger data centers and data centers that plan to scale usually implement one or more HDAs. Aside from scale, HDAs can make it easier and less risky to make cable-related changes, as they often allow you to reduce the number of cabling changes required within the MDA.

Horizontal Distribution Area

The horizontal distribution area (HDA) is optional. If your data center is large or you plan to scale it to become large, you'll want to implement one or more HDAs. If you expect your data center to always remain small, you can skip the HDA and cable directly from the MDA to your various EDAs.

NOTE

When talking about the size or scale of your data center, it isn't always high port count (a high number of network ports in the data center) that dictates the necessity of implementing HDAs. HDAs can also be useful if the distance from the MDA to your various EDAs exceeds recommended cable lengths. This is because the switching technologies employed in the HDA are *active,* meaning that they repeat the signal. *Repeating* a signal allows signals to be transmitted over large distances.

If you implement HDAs in your data center, this is sometimes where your core SAN switches may be placed, as they act as consolidation points between storage arrays and edge switches that the servers connect to. However, it is also common to install SAN switches in the EDAs of your data center. In IP networking, the HDAs usually map to the *aggregation layer*.

Cabling between the MDA and the HDA is backbone cabling.

NOTE

Very large data centers can also implement zone distribution areas (ZDAs) that sit between the HDA and EDA and act as consolidation points to potentially allow for more flexibility.

Equipment Distribution Area

The equipment distribution area (EDA) is where most of the server and storage action happens, as this is where servers, storage, and top-of-rack (ToR) network switches live.

The cabling between HDAs and EDAs is considered *horizontal cabling*. Although in the past copper was king in horizontal cabling, it is fast losing popularity to fiber cabling, because of the ability for fiber cabling to deliver 40G Ethernet, as well as being road-mapped to handle 100G Ethernet.

Copper twinax direct-attach cable (DAC) is a popular choice when cabling within the EDA, especially cabling within a single rack. For example, copper twinax is commonly used for cabling between servers and ToR switches, as it is capable of transmitting 10G Ethernet over distances of up to 5 meters. Copper twinax solutions are also more power-efficient than other copper alternatives such as unshielded twisted-pair (UTP) and shielded twisted-pair (STP), due to the fact that they use more power-efficient PHY chips in the server and switch.

Aside from the aforementioned concepts, the TIA also recommends using the highest-capacity cabling media possible—such as laser-optimized OM4 and the highest category specification available—in order to future-proof the data center and mitigate potential future disruption. This is sound advice. Think long and hard about the life expectancy of your cabling.

Working in the Data Center

There are several rules and courtesies that will help you with any work that you carry out in a data center.

Access Control

The levels of access control differ depending on the size and nature of the data center. As a general rule, only authorized personnel should be permitted access to your data centers. Even you, as an IT/storage admin, might not be permitted uncontrolled access to the data center. It is common practice for data centers to have high fences with security gates manned by dedicated security staff. To get through the front gate, you normally need to be preauthorized.

Real World Scenario

Date Center Security Should Be Taken Seriously

The CIO of a major global financial institution turned up unannounced at one of the company's major data centers. Because his name was not on the list of people authorized to access the data center that particular day, he was denied access.

Once through the security gate, it is normal practice for access to the ER and computer rooms to be secured with electronic access systems such as swipe-card access or biometric access. Further, some data centers secure their servers, network, and storage equipment inside locked cabinets with badge or pin-code access. Opening the cabinets requires preauthorization.

It is also a common practice for the ER to be outside the computer rooms so engineers from your network providers don't need to access your computer rooms.

Fire-Prevention Systems

Obviously, a fire in the data center could be catastrophic to your organization. Data centers need best-of-breed fire-prevention systems that include the following:

- Fire warning systems
- Fire suppression systems

When it comes to fire suppression, data centers have systems ranging from small handheld fire extinguishers all the way up to expensive, top-of-the-line, gas-based fire suppression systems. If possible, water-based fire suppression systems should be avoided, as water and computer equipment doesn't usually mix well! However, gas suppression systems, which control and put out fires by starving them of oxygen, can be expensive.

General Rules of Conduct

Food and drink should be banned in all data centers. Gone are the days of IT guys resting a half-full beverage on top of a server while they work at the console. In fact, in many data centers, if you're caught with food and drink, you will be immediately walked off-site and asked not to come back.

It's also good manners and a best practice to clean up after yourself. That means leaving things in the same or better condition than they were in when you arrived. Don't leave any empty cable bags, spare screws, or anything else behind after you leave. Always remember to put floor tiles back safely and securely! After all, the raised floor is often important in maintaining proper airflow in the data center.

Real World Scenario

Take Care When Working with Raised Floors

A worker in a small computer room with a raised floor lifted all the tiles in a single row in order to easily lay some cabling in the floor void. What he didn't realize was that the raised floor was substantially weakened by having an entire row of tiles lifted. As he was working with the row of tiles removed, part of the floor collapsed, damaging cables and equipment in the floor void, which required major engineering work to fix and recertify the floor.

Make sure that you carry out only authorized work while in the data center. Don't get overexcited and decide to do other jobs while you're there. For example, don't lift floor tiles without authorization. Most data centers will be equipped with closed-circuit security cameras to monitor what goes in and out of the data center, as well as what happens while people are inside.

There is one final and important point to keep in mind. Don't press the big, red, emergency power-off (EPO) button! Most data centers have a big, red, EPO button that is used to cut power to the data center in the event of an emergency. Although it can be extremely tempting to see what happens if this button is pressed, unless there is a genuine emergency, do not press it!

Summary

In this chapter we learned about the importance of the data center, and that not all data centers are created equal. We learned about the widely accepted data center tier model maintained by the Uptime Institute, and that tier 4 data centers are the highest tier and provide the best availability and redundancy. We also learned about data center racks, raised flooring, air-flow and cooling, and power. We also learned about the vital nature of cabling, and how it influences reliability of the data center today, as well as future proofing it for new technologies. We finished the chapter by talking about physical data center security, fire detections and suppression systems, and some general rules of conduct that should be adhered to in all data centers.

Chapter Essentials

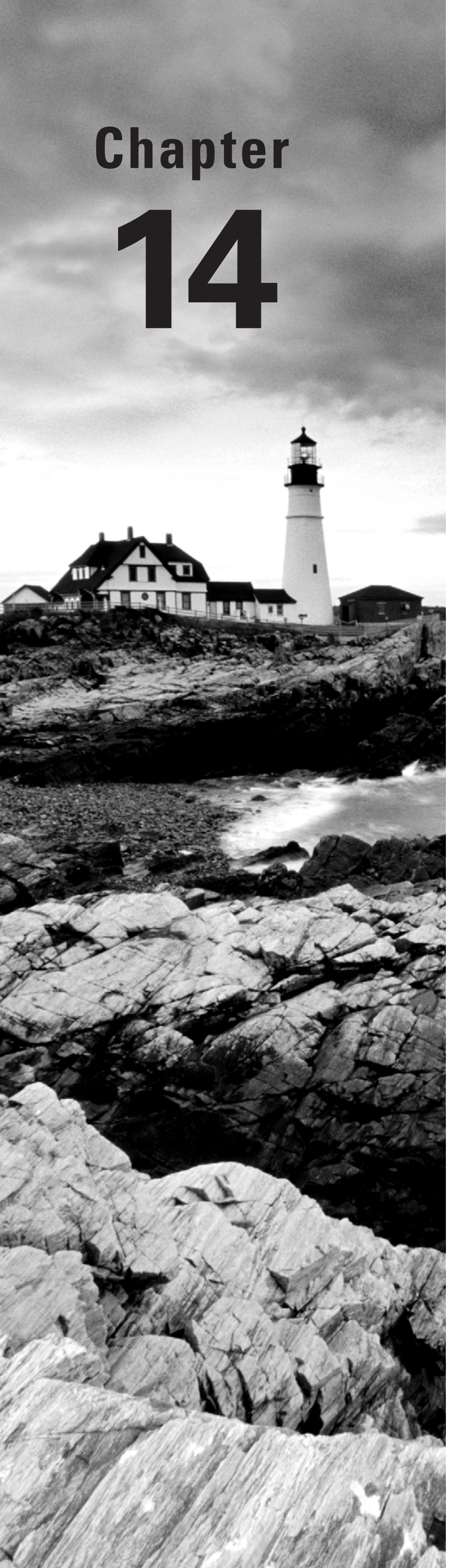
Racks Data center racks come in standard sizes, should be grounded, and need to be installed in a planned layout to facilitate efficient air-flow and cooling within the data center. Racks that come in non-standard sizes should be avoided where possible, and all work within a rack should be approved and carried out with the utmost care.

Power Modern data centers consume huge amounts of electricity and often have backup generators on-site that can provide electricity in the event that utility power is lost. Most racks and rows in the data center have N+1 redundant power supplies.

Cooling Modern IT equipment kicks out a lot of heat, and if not kept cool will malfunction. Many data centers operate hot/cold aisles and sometimes even enclose aisles to stop hot and cold air from mixing. Keeping hot and cold air separate can drastically reduce the cost of keeping the data center cool.

Structured Cabling Cabling is a vital component of the data center. Don’t cut corners when it comes to structured cabling, and make sure that you plan and deploy a cabling solution that scales well and will suit the needs of your data center 10 years from now.

Emergency Power-Off Button Every data center has at least one really cool-looking emergency power-off (EPO) button. In the event of a real emergency, this button can save lives by cutting all power to the computer room. Use it only in a real emergency. If you use the EPO button inappropriately, you should expect to be sacked.


Chapter

14

Converged Networking

TOPICS COVERED IN THIS CHAPTER:

- ✓ **The two dominant data center networks**
- ✓ **Storage network requirements**
- ✓ **Data center bridging/converged enhanced Ethernet**
- ✓ **Lossless networks**
- ✓ **Congestion notification**
- ✓ **Enhanced transmission selection**
- ✓ **Fibre Channel over Ethernet**

This chapter covers the two dominant networks that exist in just about all data centers: Ethernet and Fibre Channel. You'll learn how you can build more-efficient data centers by collapsing these two networks together. You'll also learn about the specific characteristics that storage traffic requires from a network, and how traditional Ethernet fell short of these requirements. Then you'll see the major changes that were implemented in a new version of Ethernet that is variously called either data center bridging (DCB) or converged enhanced Ethernet (CEE). This chapter also discusses how Fibre Channel frames are encapsulated and transported over these new DCB/CEE networks. And the chapter concludes with how this impacts storage network design and how it is being implemented in the real world.

A Tale of Two Networks

Most data centers have at least two major networks:

- Ethernet network
- Fibre Channel network

Let's take a quick look at each of them.

Ethernet

Ethernet is the general-purpose, jack-of-all-trades network. It is the king of both versatility and scalability and is the most commonly deployed network in the world. In fact, you're unlikely to see any data center on the planet that doesn't deploy at least one Ethernet network. Most data centers deploy multiple.

On the downside, Ethernet networks aren't deterministic when it comes to performance, particularly latency. However, the recent 10G and 40G Ethernet technologies have gone some way toward addressing the performance issues of Ethernet.

Ethernet is also what we call a lossy network. *Lossy* tells us that when the going gets tough, Ethernet starts dropping frames. It does this during conditions such as congestion, when a switch is receiving frames but can't forward them yet because the next hop is busy. When packet loss like this happens, it's the job of protocols higher in the stack, such as Transmission Control Protocol (TCP), to take care of retransmitting frames that never reach their destination.

NOTE It is important to understand how terminology is used, both in this chapter and in the real world. *Dropping* packets is sometimes referred to as *discarding* packets. And lossy networks are often referred to as being *unreliable networks*. We also often interchange the terms *frame* and *packet.* Some people will argue that frames are layer 2 constructs, whereas packets are layer 3 constructs, but for our purposes both terms mean the same thing.

In summary, while not the most performant network technology in the world, Ethernet is almost certainly one of the most versatile and widely deployed.

Fibre Channel

Fibre Channel (FC) networks are high-speed, low-latency networks that are almost exclusively used for transporting storage traffic. It's extremely rare to see Fibre Channel used for anything other than storage networking. FC also operates link-layer signaling that communicates buffer credits and status, enabling it to be a *lossless* networking technology, meaning it doesn't drop packets as a result of congestion. This combination of low latency and lossless capability makes FC ideal for transporting SCSI traffic that doesn't deal well with unreliable transports.

NOTE At a high level, when we talk about link-layer technologies, such as buffer credit systems used to ensure lossless networks, we're talking about point-to-point technologies and not end-to-end technologies. By point-to-point, we mean between two connected devices such as an HBA-to-switch connection or a switch-to-switch connection with no other devices in between, whereas end-to-end technologies are usually initiator-to-target with potentially many devices in between.

On the downside, FC networks are less versatile and less scalable than Ethernet networks, not to mention more expensive. But they are reliable. In fact, the word *channel* in *Fibre Channel* is significant. The term comes, at least in part, from the fact that SCSI is a form of channel technology, and channel technologies are a bit different from network technologies. The main goal of channel technologies is to provide high-performance, low-overhead interconnects, whereas the main goal of networking technologies is often to provide versatility and scalability. Fibre Channel is obviously an attempt at providing both low latency and scalability. It does a decent job, but it is nowhere near as scalable as a pure networking technology like Ethernet. Ultimately, FC provides a high-speed, low-latency storage network (a channel) that is relatively contention free.

Single Converged Data Center Network

The existence of two physically and logically separate networks—Ethernet and Fibre Channel—creates a good opportunity for consolidation. That's exactly what *converged networking* is all about: reducing cost through infrastructure rationalization and consolidation. The following are some of the obvious benefits of consolidating Ethernet and Fibre Channel networks:

- Reducing the number of network adapters in physical servers
- Reducing the number of cables coming out of each physical server
- Reducing the number of switches in the infrastructure

So instead of having one set of network adapters, cables, and switches dedicated to Fibre Channel and another set of adapters, cables, and switches dedicated to Ethernet, with a converged network we have only a single set of network adapters, cables, and switches and run our Ethernet and FC storage traffic over them. That pretty much promises to make the *wire once* ideal a reality. You just rack a server on day one with a converged network adapter (CNA) in it, and you can do any and all the networking you want. You can do general-purpose IP networking, FC storage networking, iSCSI storage, NAS, maybe even low-latency, high-performance computing. A single network adapter and a single cable will do it all!

These reductions have benefits, especially in large data centers and data centers deploying densely packed servers. As data center racks become more and more densely populated with servers and storage, the prospect of reducing the number of network adapters, cables. and switches suddenly becomes very appealing. It also has the positive effect of reduced power and cooling costs. All in all, it delivers reduced data center running costs and lower total cost of ownership (TCO).

However, as is so often the case, it's not as simple as just slapping your FC storage traffic on any old Ethernet network. There are specific requirements that, if not addressed, will make your life as a storage administrator very difficult. Let's take a look at these requirements.

Storage Network Requirements

It's easy to forget that the main job of FC, as a storage networking technology, is to facilitate the reliable and efficient transportation of SCSI commands between initiators and targets. It also emulates, as much as possible, the old internal SCSI bus that is what the SCSI protocol was originally designed to work with.

Let's recap a little on SCSI. SCSI (Small Computer System Interface) was originally designed to be used within the confines of a physical server chassis, running uncontested over relatively short parallel cables. *Uncontested* indicates that no other protocols or traffic are contending for the same bandwidth. There is literally zero contention. As a result, SCSI was not designed to deal well with delays, congestion, or transmission errors. In fact, when these occur, SCSI deals with them very poorly. So the aim of the game is to avoid them at all costs.

One of the reasons that Fibre Channel works so well at transporting SCSI traffic is that it possesses certain features and characteristics that provide an experience similar to a local SCSI cable inside a server chassis. Some of these features and characteristics include the following:

- Simple topologies
- Good bandwidth
- Low latency
- Reliability (lossless)
- Deterministic performance

Traditional Ethernet, on the other hand, isn't such a good match for SCSI. To start with, congestion is commonplace on Ethernet networks. When contention is present, you can kiss goodbye any hopes of deterministic performance or low latency. As mentioned earlier, Ethernet networks are also lossy—they drop frames when congestion occurs. None of this bodes well for SCSI.

In summary, you certainly won't be simply wrapping your FC frames inside regular Ethernet frames and dropping them on your existing gigabit Ethernet network. Clearly something has to change in order for Ethernet to be a viable transport for FC storage traffic!

Enhanced Ethernet

In order to create a single data center network capable of transporting IP and FC storage traffic, Ethernet had to be significantly enhanced and upgraded. These changes are so extensive that many people feel it barely resembles the Ethernet that most of us cut our networking teeth on.

NOTE

Sometimes we refer to the *data center network* as a *unified fabric*. Both terms mean basically the same thing: a single network that can transport LAN and SAN traffic.

To create this new *enhanced Ethernet,* the Institute of Electrical and Electronic Engineers (IEEE) formed a new task group within the 802.1 working group. This new task group is called the Data Center Bridging (DCB) task group and is responsible for the development of a data center Ethernet network that is capable of transporting all common data center network traffic types:

- IP LAN traffic
- FC storage traffic
- InfiniBand high-performance computing traffic

Depending on who you speak to, this enhanced Ethernet is usually called either *data center bridging (DCB)* or *converged enhanced Ethernet (CEE)*. We also sometime call it a *data center fabric* or *unified fabric*.

If CEE wants to transport FC storage traffic, it needs a bunch of enhancements, including the following:

- Increased bandwidth
- Classes of service
- Priorities
- Congestion management
- Enhanced transmission selection (ETS)

Throw on top of these logical changes a whole load of new hardware requirements—cables, network adapters, switch ports, and switches—and you could be forgiven for wondering whether it's worth it. Let's go through the individual enhancements that we previously listed.

Increased Bandwidth

In order to meet the bandwidth and other demands of a unified data center fabric, CEE is a collision-free, full-duplex network that operates at a minimum of 10 Gbps. This is vital if we want to consolidate multiple 1 Gbps LANs and 2 Gbps and 4 Gbps SANs onto a single wire. 10G CEE can also safely transport 8 Gbps FC traffic, as well as allowing the entire 10 Gbps of link bandwidth to be used by the latest-generation CNAs.

10 Gbps enhanced Ethernet is only the start. 40 Gbps Ethernet and 100G Ethernet are already here, and in time both will be affordable. A strong technology road map like this positions Ethernet well as the data center network of the future.

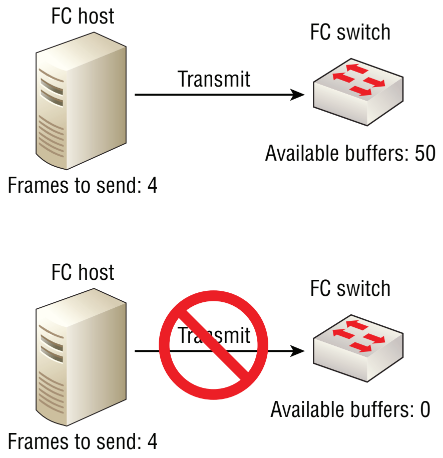
Priorities and Classes of Service

I mentioned earlier that FC storage traffic requires a lossless network. But what exactly is a lossless network? Stated very simply, a lossless network doesn't drop frames because of congestion. This is the opposite of a lossy network, which does drop frames, usually when congestion occurs. Consider the following high-level and oversimplified example: A switch receives 100 frames but is unable to immediately forward them because the switch at the next hop is saturated and unable to accept more frames. As a result, the switch buffers the 100 frames, and congestion arises. After the switch's buffers are full, it can't accept any more frames and starts to discard (drop) them.

The way that networks generally implement lossless behavior is via link-layer signaling that keeps track of buffers at the remote end. We usually call this *flow control*. As an example, in the FC world, flow control is implemented via a link-layer system called buffer-to-buffer (B2B) flow control, where the sender is not allowed to send frames unless it explicitly knows that the receiver has enough buffers to receive the frames. It's simple and effective at ensuring that congestion does not arise and that packets are not dropped. FC flow control is shown in Figure 14.1.

The Data Center Bridging task group has decided to implement link-layer flow control in CEE via a mechanism called *priority-based flow control*, or PFC for short.

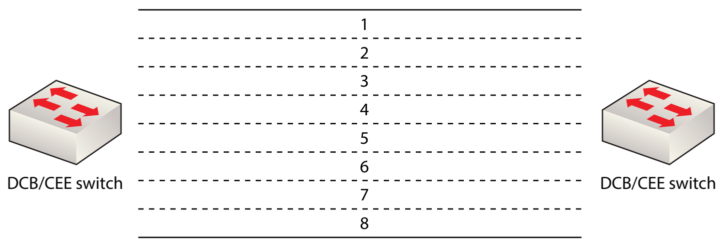
FIGURE 14.1 Fibre Channel B2B flow control



NOTE In some older documentation, you may find PFC referred to as *per priority pause* or *class-based flow control*.

Before we dig into PFC, it is worth taking a moment to briefly talk a little about *Ethernet priorities*. CEE defines eight priorities that allow for eight classes of service at the link layer. This is done by tagging frames with an encoded priority, effectively allowing the bandwidth of a CEE network to be divided into eight logical lanes, or virtual links. Figure 14.2 shows a link between two CEE switches divided into eight logical lanes, labelled 1 through 8.

FIGURE 14.2 CEE network divided into eight logical lanes

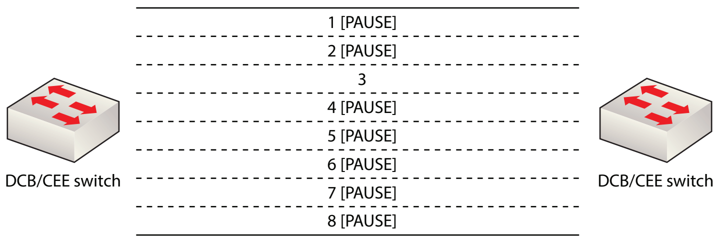


Now that you understand about the eight priorities in a CEE network, let's take a look at how PFC works. PFC leverages these eight logical lanes (priorities) and can selectively enforce a PAUSE condition for each of the lanes. This done by issuing a special PFC PAUSE frame on the network, specifying the following:

- Which of the lanes the PAUSE condition applies to
- How long the PAUSE condition is to last

By issuing PAUSE frames that instruct all but one lane to stop sending, you have effectively prioritized the lane that has not been paused. Likewise, if you enforce the PAUSE condition on a single lane, you have effectively de-prioritized that lane and prioritized all other lanes. Figure 14.3 shows the eight logical lanes of CEE with all lanes paused except lane 3.

FIGURE 14.3 PAUSE condition applied to all lanes except lane 3



It is also possible to selectively remove the PAUSE condition, such as if the congestion has dissipated sooner than expected, and there is no need to wait until the pause time-out expires.

Congestion Management

Congestion management is a way of trying to avoid network congestion, hopefully at the source, before it starts causing problems on the network. Common approaches to congestion management include rate-limiting and selectively pausing frame transmission. However, trying to solve the problem of congestion in large networks is a huge task and one that's fraught with danger. It is possible for congestion management systems to make things worse. To date, none of the attempts at standardizing congestion notification have been overly popular.

Enhanced Transmission Selection

Enhanced transmission selection (ETS) is an attempt to standardize quality of service (QoS) on CEE networks. ETS allows network bandwidth to be sliced and diced among the eight classes of service defined by CEE. This bandwidth allocation is *dynamic*, meaning that if a particular class of service isn't using its full allocation of bandwidth, its spare allocation is available for other classes to borrow. However, as soon as the allocated bandwidth is required by the class to which it is assigned, other classes that are borrowing bandwidth have to give it back.

This dynamic allocation and reallocation of bandwidth allows for intelligent use of available resources and is a huge improvement over previous models in which different classes of traffic had their own fixed bandwidth allocations with no ability to share.

Data Center Bridging Capability Exchange

With all the new features and configuration options that come with CEE and FCoE (which we'll talk about in the next section), life would be a lot easier if there was a simple, automated, and effective mechanism for negotiating and configuring these options. For example, as new devices are added to a CEE network, it would be great to be able to automatically negotiate which capabilities and options this newly added device supports. It would also be great to have the ability to push standard configurations to these newly added devices. With all of this in mind, Data Center Bridging Capability Exchange Protocol (DCBX) was devised.

Devices can use DCBX to issue configurations to other devices in the network such as CNAs and other switches. This can make network administration simpler by allowing for central management of configurations as well as reducing the chance of misconfiguration.

NOTE DCBX is a standards-based extension of the Link Layer Discovery Protocol (LLDP).

PFC and ETS are both examples of link-layer features that can be negotiated and configured via DCBX. After all, there wouldn't be much point in a switch issuing PFC configuration and instructions to a newly attached CNA if the CNA did not support or understand PFC.

Fibre Channel over Ethernet

Fibre Channel over Ethernet (FCoE) is literally what it says on the package: Fibre Channel frames delivered over an Ethernet network. To put it in slightly more technical terms, Fibre Channel frames are statelessly encapsulated within layer 2 Ethernet frames and transported over an IEEE 802.3 wired Ethernet network.

How do Fibre Channel and Ethernet interact with each other? As far as Ethernet is concerned, FCoE is no different from any other upper-layer protocol (ULP) such as IP or MPLS. As such, Ethernet is happy to transport FCoE traffic. On the flip side, as far as Fibre Channel is concerned, Ethernet is just a new physical transport medium, replacing the lower layers of the FC stack, FC-0 and FC-1.

NOTE The FCoE standards are driven by the International Committee for Information Technology Standards (INCITS) T11 technical committee. This T11 technical committee is the same one that defines the FC standards.

In order for an FC frame to be delivered over an Ethernet network—remember, it has to be a CEE Ethernet network—the FC frame has to be encapsulated within an Ethernet frame. Let's take a look at encapsulation.

FCoE Encapsulation

Each and every FC frame is slightly more than 2 KB in length. However, standard Ethernet frames are only 1.5 KB in length. It doesn't take a rocket scientist to figure out you can't fit 2 KB inside 1.5 KB. So something needs to be done. There were two obvious options available to the guys defining the FCoE standards:

- Fragment FC frames
- Use larger Ethernet frames

From a performance and simplicity perspective, fragmenting frames should be avoided at all costs. Fragmentation adds complexity and overhead to the encapsulation process that ultimately results in latency and lower performance. That's not good for FC storage traffic.

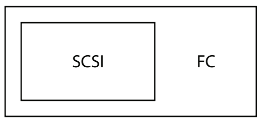
Using larger Ethernet frames requires the use of an Ethernet technology called *jumbo frames* that allows Ethernet frames to be up to 9 KB in size. Clearly, jumbo frames up to 9 KB are more than capable of encapsulating 2 KB FC frames without having to resort to fragmentation. This allows the encapsulation process to be kept as simple, lightweight, and fast as possible—one FCoE frame for every FC frame. There is no fragmentation and reassembly overhead, so there is no unnecessary protocol-related latency. All of this is important for transporting FC storage traffic. So for these reasons, the standards body decided on option 2, using jumbo frames.

NOTE The Storage Networking Industry Association (SNIA) defines *baby jumbo frames* as frames that are 2.5 KB in length and states that FCoE encapsulation requires *at least baby jumbo (2.5KB) Ethernet frames.*

Now let's look a little closer at the FCoE encapsulation process.

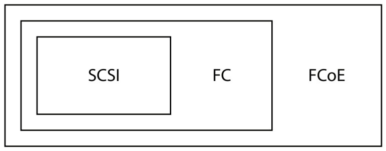
FC exists to transmit SCSI data, and SCSI commands and data are already encapsulated within FC frames, as shown in Figure 14.4.

FIGURE 14.4 SCSI encapsulated within FC



FCoE takes this encapsulation one step further by adding a layer of encapsulation—encapsulating FC frames within FCoE frames, as shown in Figure 14.5.

FIGURE 14.5 FC encapsulated within FCoE



This encapsulation of FCoE frames is a hop-by-hop process, meaning that each and every FCoE switch in the network has to strip off the FCoE encapsulation on receipt of an FCoE frame and then re-encapsulate the frame before forwarding it on to the next hop. If this isn't done quickly and efficiently, it will quickly start to impact performance. For this reason, CNAs perform this encapsulation in hardware on an ASIC, as do most FCoE switches.

NOTE There are software FCoE stacks out there, allowing you to do FCoE without needing expensive specialized CNAs. Obviously, this design requires the use of host resources such as CPU cycles to perform the FCoE processing. Although iSCSI has very much gone down the route of the software iSCSI initiator rather than the hardware iSCSI initiator, iSCSI is almost always deployed on a far smaller scale than FC (and consequently FCoE). While not necessarily recommended for production use cases, especially with high performance requirements, a software FCoE stack can be useful for labs and test environments. In vSphere 5, VMware introduced a software FCoE adapter that works with certain certified 10G cards. This can be a great way of playing around with FCoE in the lab, as long as you have FCoE-capable switches in the lab.

There is one final but critical point regarding FCoE encapsulation. The FCoE encapsulation process leaves the FC frame intact. Things like Source_ID (S_ID), Destination_ID (D_ID), and world wide port names (WWPNs) are untouched. This means that FCoE doesn't do away with existing FC concepts and mechanisms such as zoning, WWPNs, and the Simple Name Server. These all remain unchanged, allowing us to continue using the familiar FC zoning concepts we've been using for years.

NOTE It's worth explicitly pointing out that FCoE does not involve IP. In FCoE environments, FC frames are not encapsulated within IP packets. FC frames are encapsulated within layer 2 Ethernet frames, meaning that FCoE networks are large, flat, layer 2, nonroutable networks. FCoE frames cannot be routed by IP at layer 3. This is in contrast to iSCSI, which is SCSI encapsulated within IP packets and is routable at layer 3 by IP.

Converged Network Adapters

Traditional Ethernet networks are accessed via a network adapter called a *network interface card (NIC)*. Each host that wants to connect to the Ethernet network needs at least one. Conversely, traditional Fibre Channel networks are accessed via a network adapter called a *host bus adapter (HBA)* in each host. Accessing an FCoE network requires a new type of network adapter called a *converged network adapter (CNA)*. All three of these network adapter cards are implemented as PCI adapter cards. They can be either expansion cards or directly on the motherboard of a server in what is known as *LAN on motherboard (LOM)*.

A CNA is exactly what the name suggests: a NIC and an HBA converged into a single network card. For performance reasons, CNAs provide NIC and HBA functionality in hardware (usually an ASIC) so that things like FCoE encapsulation can be fast without impacting host CPU resources.

Now let's look at how CNAs enable infrastructure consolidation. In a traditional data center with distinct Ethernet and FC networks, it's not uncommon for physical servers to have at least the following network cards:

- Two 1 Gbps Ethernet NICs for your production network
- Two 1 Gbps Ethernet NICs for vMotion
- Two 8 Gbps FC HBAs for storage
- Two 1 Gbps Ethernet (dedicated for management traffic)

Eight network cards obviously require eight cables. That's just cables. Having independent Ethernet and FC networks requires a lot of networking hardware too. You need dedicated Ethernet switches and dedicated FC switches. However, with a converged data center network, the number of network cards in the same server can easily be reduced as follows:

- Two 10 Gbps CNA
- One 1 Gbps NIC (dedicated for management traffic)

The two 10 Gbps CNAs can handle all IP and FC traffic requirements.

In order to converge networks, one concern that you may need to get over is the notion of sharing a single CNA for multiple functions. Some organizations have strict policies over network adapter isolation, where a dedicated network adapter card is used for each application. So a server with two applications would have two NICs, plus a separate NIC for management, and so on. This mindset will have to be overcome if you are to converge networks, as you'll be running FC and IP over the same network adapter!

Of course, CNAs tend to be more expensive than NICs, though not too dissimilar to HBAs.

FCoE and Lossless Network Capability

As FCoE is carrying encapsulated FC frames, it requires a lossless network. If the network/transport isn't lossless, recovery and retransmission would have to be handled by SCSI, and that wouldn't be good. SCSI was never designed to be good at recovery.

As mentioned earlier in the chapter, CEE defines eight logical lanes that allow us to create eight potential classes of service at the link layer by tagging frames with an encoded priority. For example, FCoE frames can be tagged with a priority of 3, effectively assigning FCoE frames to lane 3, or class of service 3. This priority can then be utilized by CEE technologies such as PFC to ensure that FCoE frame delivery can be prioritized or de-prioritized when congestion occurs.

NOTE Setting a priority (class of service) is usually done by setting the service priority bit in the VLAN tag.

Let's look at a quick example. Figure 14.6 shows a network with two FCoE switches. The link between the two FCoE switches is divided into eight lanes (priorities), and the network is experiencing congestion as a result of extremely high traffic over lane 2. As you can see in Figure 14.7, the network has issued a PFC PAUSE condition onto lane 2—effectively de-prioritizing it, and prioritizing all other seven lanes. This has the benefit of allowing our FCoE frames on lane 3 to continue being delivered and gives the network a chance of recovering from the congestion.

FIGURE 14.6 PFC and eight logical lanes

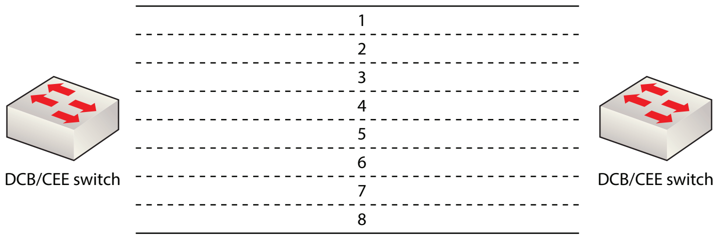
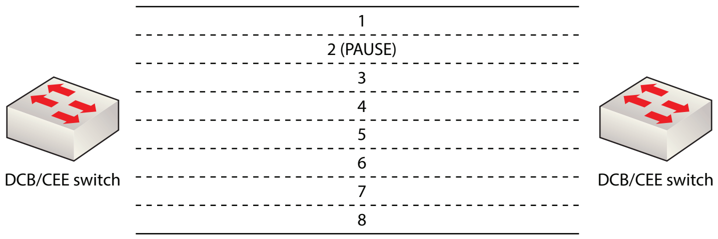


FIGURE 14.7 PFC PAUSE condition imposed on lane 2



In Figure 14.7, PFC selectively applies the PAUSE condition to lane 2. This PAUSE condition tells lane 2 to stop transmitting frames for a certain period of time. Once this period of time has elapsed, frames can again be transmitted on lane 2. It is also possible for the network to remove the PAUSE condition if the congestion dissipates before the PAUSE has expired.

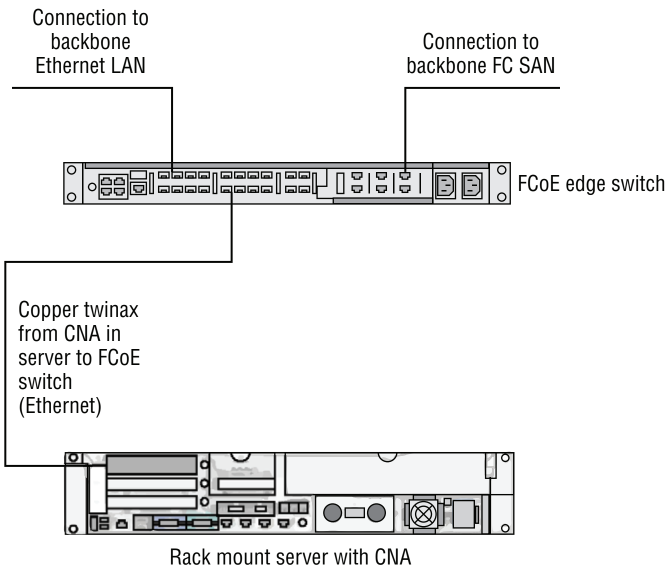
NOTE Although CEE contains specifications for congestion notification, this is not needed for FCoE. In an FCoE network, flow control is handled by PFC and not FC buffer credit systems.

FCoE Switches

With all of the new technologies in CEE and FCoE, it is inevitable that new switches will be required. Fortunately, there are plenty already available in the market from several vendors.

An FCoE switch connects to the LAN and the SAN. It has both CEE ports and FC ports. This is useful for companies that already have significant investment in FC SAN technology, as it allows FCoE switches to be seamlessly added into the existing SAN environment. Figure 14.8 shows an FCoE switch connected to both the corporate LAN and corporate SAN.

FIGURE 14.8 FCoE switch connecting to legacy FC devices


Connection to backbone Ethernet LAN
Connection to backbone FC SAN
FCoE edge switch
Copper twinax from CNA in server to FCoE switch (Ethernet)
Rack mount server with CNA

Although FCoE frames are transported over CEE networks, the forwarding of FCoE frames is not the responsibility of Ethernet forwarding. This means that the typical Ethernet layer 2 multipathing technologies such as Spanning Tree Protocol (STP), EtherChannel, and Transparent Interconnection of Lots of Links (TRILL) have no influence over the forwarding of FCoE frames. Instead, FCoE forwarding is the responsibility of a function in the FCoE switch known as the FCoE Forwarder (FCF). The FCF forwards FCoE frames based on the FC Destination_ID (D_ID) field of the encapsulated FC frame, as well as a routing table created by the Fabric Shortest Path First (FSPF) protocol.

FCoE in the Real World

As with most technologies, people like to kick the tires a little before deploying them in their production estates that support their most important applications and data. Also, as IT budgets are getting tighter and tighter, most organizations simply don't have the capital to be able to rip and replace existing Ethernet and FC networks with shiny new FCoE networks—even if they bring long-term cost benefits. As a result, uptake of FCoE in the real world has been lukewarm at best.

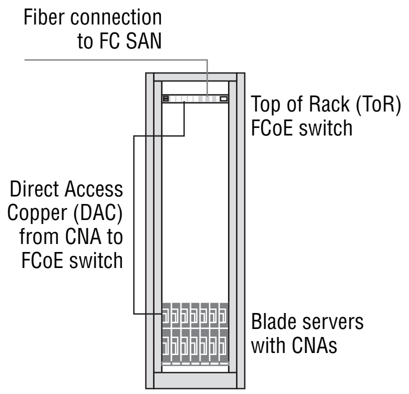
Those who have deployed FCoE have usually deployed it in either or both of the following two ways:

- As part of a prepackaged converged computing network and storage package
- At the access layer (the first hop between host and access layer switches, such as Top of Rack)

Prepackaged products include so-called *converged blocks* or *converged pods*, such as VCE Vblock or NetApp FlexPod. In these packages, you buy the entire stack of computing, network, and storage as a single packaged product. These blocks tend to come with servers that have CNAs and boot from FCoE SAN storage. You often don't have the choice of how to configure them and are forced to deploy CNAs in your servers and FCoE switches at that first hop from the server to the Top of Rack switch (not that this pre-scripted design is always a bad thing).

The other major area where people are deploying FCoE is at the access layer of the network. This means the first hop onto the network, between the network adapter in the server and the access-layer network switch that usually sits at the top of the same rack in which the server is installed. This is shown in Figure 14.9.

FIGURE 14.9 FCoE at the network access layer


Fiber connection to FC SAN
Top of Rack (ToR) FCoE switch
Direct Access Copper (DAC) from CNA to FCoE switch
Blade servers with CNAs

The access layer of the network is a natural and good place to start deploying any new technology. For starters, it's far less risky to deploy a new technology at the network access layer than it is to drop it right in at the core. Deploying at the access layer allows you to slowly and systematically deploy FCoE—one rack at a time—and gain confidence in the technology before starting to deploy deeper within the network.

As a result, most deployments of FCoE, other than the likes of NetApp FlexPod and VCE Vblock, tend to be at the network access layer by deploying new servers with CNAs instead of NICs and HBAs. This reduces the network-related cabling and the number of network switches in the rack. Deploying CNAs in your servers is also future-proofing your servers against any future FCoE deployments, as the CNAs can start today doing just 10G Ethernet, but if you deploy FCoE in the future, you can also deal with FCoE.

Another consideration when deploying FCoE in the real world relates to who owns the converged infrastructure. Who owns the CNAs, and usually more important, who owns the FCoE switches? The following should be considered when making this decision:

- Who decides which FCoE switches to purchase—vendor and model?
- Who pays for the switches and the support?
- Who decides when and what versions of firmware to deploy?
- Who has root access to switches?

These are not always easy decisions. In most organizations, the trend seems to be toward the server team owning the CNAs, and the network team owning the FCoE switches, with the storage guys providing the expert advice on the configuration of CNAs and FCoE and FC settings within the FCoE switches.

Although FCoE is somewhat complementary and similar to server virtualization, it has to be pointed out that uptake of FCoE has been nothing like the uptake of server virtualization. This is in no small part because FCoE hasn't brought anywhere near the kind of flexibility and agility that server virtualization has. In fact, if anything, software-defined networking (SDN) looks more likely to bring the kind of revolutionary flexibility and new models similar to those brought with server virtualization technologies.

Summary

This chapter covered the two dominant networking technologies seen in most data centers around the world: Ethernet and Fibre Channel. You learned about the differences between them as well as the potential benefits of combining them into a single data center network. The chapter showed how Ethernet was the natural choice, because of its wider adoption and increased versatility and flexibility. The chapter also outlined the features that were lacking in Ethernet in order to make it a viable transport for storage traffic and then went on to detail how major changes have been implemented into a new version of Ethernet—known as data center bridging (DCB), or converged enhanced Ethernet (CEE)—that make it suitable or transporting storage traffic. You then learned how FC is layered on top of DCB/CEE, and finished the chapter seeing how this impacts future network design as well as how this is being implemented in the real world today.

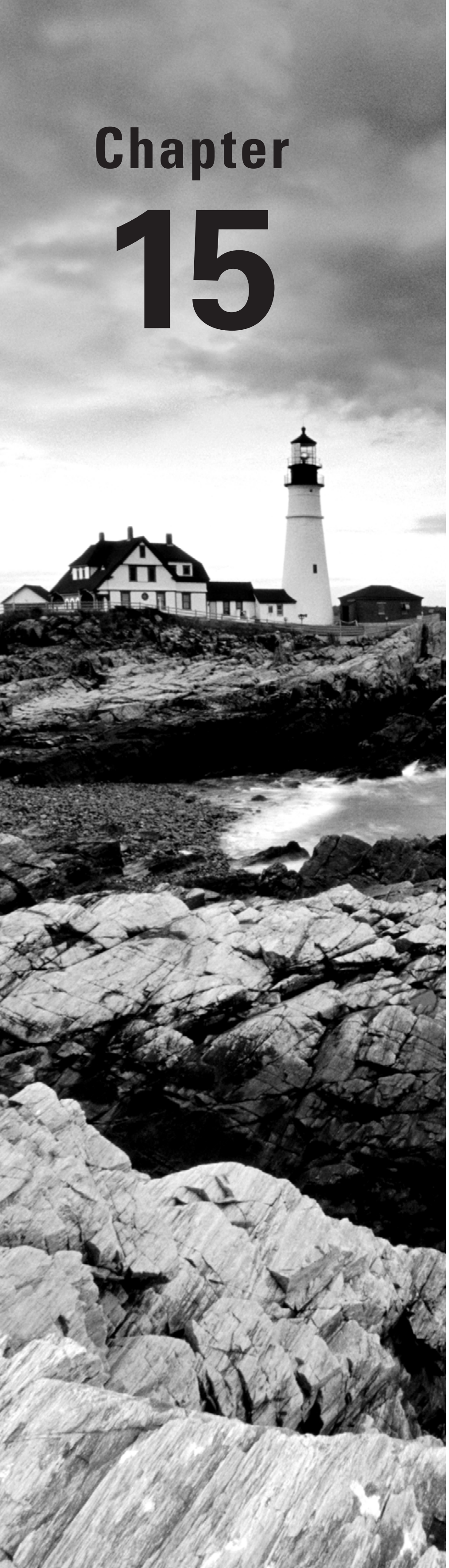
Chapter Essentials

Data Center Bridging Data center bridging (DCB) is a task group within the Institute of Electrical and Electronic Engineers (IEEE) 802.1 working group. The task group is responsible for implementing enhancements to 802.1 networks to enable them to be the natural choice for future data center networks. The version of enhanced Ethernet that the DCB task group is responsible for is sometimes called DCB Ethernet.

Converged Enhanced Ethernet Converged enhanced Ethernet (CEE) is a new enhanced version of Ethernet suitable for transporting storage traffic as well as traditional IP traffic. This makes it a natural choice for data center network consolidation.

Lossless Networks Lossless networks don't drop frames when congestion occurs. This is a key network feature for any network wishing to transmit FC traffic.

Fibre Channel over Ethernet Fibre Channel over Ethernet (FCoE) is the encapsulation of Fibre Channel frames within Ethernet frames, allowing Fibre Channel traffic to be transported over converged enhanced Ethernet networks. FCoE is an important technology in enabling data center network consolidation.


Chapter

15

Cloud Storage

TOPICS COVERED IN THIS CHAPTER:

- ✓ **Defining the cloud**
- ✓ **The cloud computing model**
- ✓ **Public cloud**
- ✓ **Private cloud**
- ✓ **Hybrid cloud**
- ✓ **Cloud storage**
- ✓ **Eventual consistency**
- ✓ **Data durability**

This chapter delves into the increasingly popular and important topic of the cloud. The chapter starts by defining the cloud at a high level and then gets into more detail. Because the concepts of cloud computing can represent a significant mindset change for many people, you'll get hands-on with some of the most popular cloud services available, including spinning up virtual machines (VMs) as well as object stores in the public cloud.

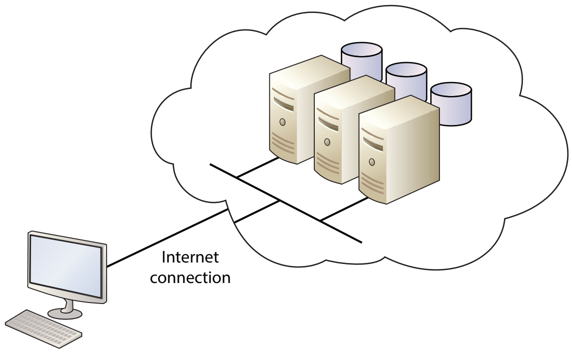
This chapter presents the three major cloud models: public cloud, private cloud, and hybrid cloud. After learning about cloud computing, you'll focus on cloud storage—its important concepts and theory as well as some hands-on activities as you create storage in the cloud and upload objects. You'll explore public cloud storage, private cloud storage, and hybrid cloud storage and some potential use cases for each.

The Cloud Overview

There isn't really any formal definition of what the *cloud* is, but don't let that deter you. The cloud is real, and it's going to get bigger in the future. Thankfully, it's pretty simple.

The cloud, or *cloud computing*, is a computing model. A major feature of this model is that IT resources are accessed remotely via the Internet, rather than locally over a local area network (LAN). This is shown in Figure 15.1.

FIGURE 15.1 High-level cloud computing model



In many ways, cloud computing is an extreme form of virtualization, where you as a consumer don't care about the underlying hardware or software. You no longer have to think about hardware maintenance, hardware upgrades, tech refresh, security patching, antivirus, support staff, or any of the typical things that you have to consider if you own the technology. You literally consume the service and don't care about the underlying specifics. It couldn't be simpler. A major goal of cloud computing is exactly that: make consumption of IT resources as simple as possible.

NOTE The term *cloud* probably came from the popular cloud image used to represent the Internet in IT and network infrastructure diagrams. One reason for representing the Internet as a cloud is that for most purposes you don't need to understand the complexities of the Internet and Internet routing; you just need to know that it's a network that reliably delivers data. The same goes for cloud computing: you pay for and receive a service and don't need to care about the details. All you care about is that it works.

Companies that provide cloud services are usually referred to as *service providers,* or *cloud providers*. They own the technology behind the service and are responsible for making sure that it all works seamlessly for you as a customer. They're also the people that you pay, usually on a monthly basis, for the privilege of using their services. For these reasons, we often refer to cloud solutions as *fully managed services*. Some of the major public cloud providers include companies such as Amazon, Microsoft, Google, and Rackspace.

One of the major advantages the cloud brings is cost. Cloud services are almost always set up so that you pay monthly based on what you have consumed that month. This operating expenditure (op-ex) model means that the financial road into the cloud is made easy. There's no requirement for large up-front capital expenditure (cap-ex) costs as there usually is in noncloud models. In noncloud models, if you want 10 servers with 64 GB RAM each and a total of 6 TB of storage, you have to lay down the cash to buy these up front (cap-ex). This isn't the case with cloud computing. In the cloud model, you can literally spin up the 10 virtual servers with the required RAM and storage (using the cloud provider's self-service tools), and then you start paying on a monthly basis. After a month or two, if you realize that you really need only three of the servers, you turn off the seven you don't want and immediately stop paying for them. Then, at a later date, if you need another 20 servers, you can spin up another 20 servers and have them instantly available. Again, there is no up-front cap-ex cost and no long lead-time on delivery of the servers.

This ability to dial up or dial down your consumption on the fly is known as *elastic computing*. You can use as much or as little of the cloud as you want, and you can change it on a daily basis. There is no need to over-spec on day one, just in case you might need more capacity in the future, only to find that that future requirement never materializes. The cloud model is totally different from more-traditional approaches to IT. It is way more flexible.

Another advantage of the cloud is that services tend to be self-service. Cloud users can order their own configurations via a simple web portal, and after they've completed their order, the service is provisioned and almost always instantly available. For example, a cloud service will probably provide a web portal for ordering a virtual machine (VM) to your exact specifications. On the portal, you enter all of the following for your VM:

- Name
- Number of CPUs
- Amount of memory
- Number and speed of network cards
- Number and size of disks

After you've ordered your VM from the self-service portal, you click Create, and the VM is created and immediately made available to you. It is a lot like buying a new book for your Amazon Kindle device—you select the book you want, add it to your shopping cart, proceed with payment, and the book is immediately delivered to your device.

Now let's talk briefly about the three major cloud models:

- Public
- Private
- Hybrid

Public Cloud

The *public cloud* is what most people think of when they think about the cloud. In many ways, it's the purest form of the cloud.

The public cloud is usually accessed over the Internet and provides the highest levels of abstraction. By this, I mean that it's opaque: customers have no visibility of the technologies holding up the service, the server hardware, and the networking and storage technologies involved.

One thing we all need to understand and accept about the public cloud is that it's *multitenant*, meaning that all the underlying infrastructure is shared by multiple customers. You definitely don't get a physical server or physical storage array to yourself! If you place your services or data in the public cloud, it will be sitting on the same shared infrastructure as potentially thousands of other customers' services and data. This is true multitenancy!

Also, as a public cloud customer, you have very little idea of where your data is actually held. Sure, you will probably know the region that it's held in, such as North America, the European Union (EU), Asia Pacific (APAC), and so on. But you won't know exactly where, and you definitely can't turn up at the front door of your cloud provider's data centers and get inside to have a look around.

All of this is important in providing a lot of the features we expect from cloud services, such as the following:

- Low cost
- Elasticity

- Massive scalability
- Self-service
- Accessibility

However, the uber-shared nature of public cloud services can be a challenge for some use cases and organizations. You might work for an organization that has strict security policies governing whether services can be hosted in the public cloud. In these situations, you may not be allowed to use public cloud services.

Services hosted in the public cloud are accessible from anywhere in the world that has an Internet connection, making public cloud–based services among the most accessible IT services in the world. You can access them at any time, from any device, in any part of the world, so long as you have an Internet connection.

This leads us to a vital point on public cloud: security! You need to be security savvy when working with public cloud services. Make sure you encrypt your data, and make sure that you, not your cloud provider, hold the encryption keys. And make sure that you mark as "public" only any data that you want to be publically viewable.

The most popular public cloud computing service is probably Amazon Elastic Compute Cloud (EC2), which is part of Amazon Web Services (AWS). Other popular public cloud computing services include Windows Azure and Rackspace Cloud Servers.

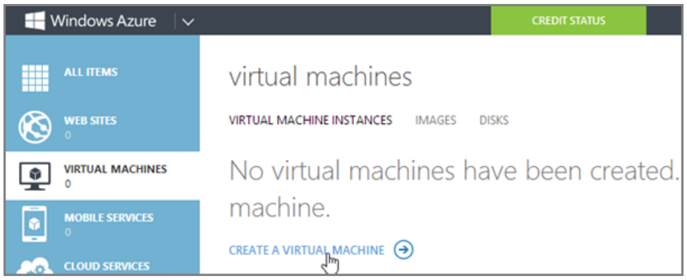
One of the first basic things you need to do in the cloud is create a virtual machine. Exercise 15.1 walks you through how to create a Windows virtual machine with Windows Azure, and Exercise 15.2 shows you how to create a Linux virtual machine with AWS.

EXERCISE 15.1

Creating a New Virtual Machine in the Cloud with Windows Azure

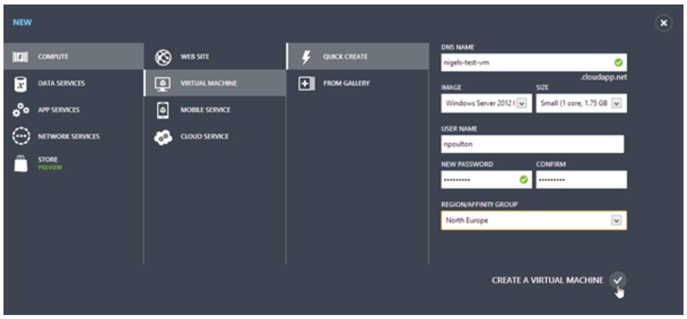
In this exercise, you'll walk through the process of creating and logging on to a Windows VM created in the Windows Azure public cloud. This exercise assumes you have a Windows Azure account.

1. In the Windows Azure web user interface, select Virtual Machines and then click the Create A Virtual Machine link.



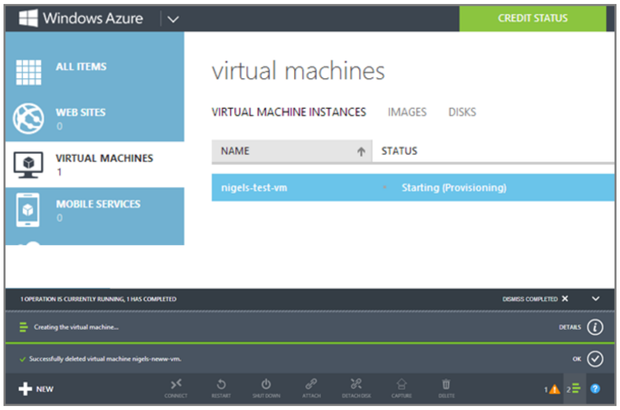
2. The properties sheet for creating a new VM appears and requires you to provide the following information for your VM:
   - DNS name
   - Operating system (referred to as *image*)
   - Size of the VM (specifies number of CPUs and amount of memory)
   - Username
   - Password
   - Region to store your VM

   Enter the information required to create your new VM and click the check mark next to Create A Virtual Machine. I am selecting a configuration for a new VM called `nigels-test-vm` that will run Windows Server 2012 on a small VM image with a single core and 1.75 GB of RAM, with a username and password, and region of Northern Europe. You should use whatever settings are appropriate for your scenario.



DNS NAME: nigels-test-vm .cloudapp.net
IMAGE: Windows Server 2012 I
SIZE: Small (1 core, 1.75 GB
USER NAME: npoulton
NEW PASSWORD: ••••••••••
CONFIRM: ••••••••••
REGION/AFFINITY GROUP: North Europe

   After you have clicked Create A Virtual Machine, Windows Azure creates your new VM. The process should take only a minute or two.



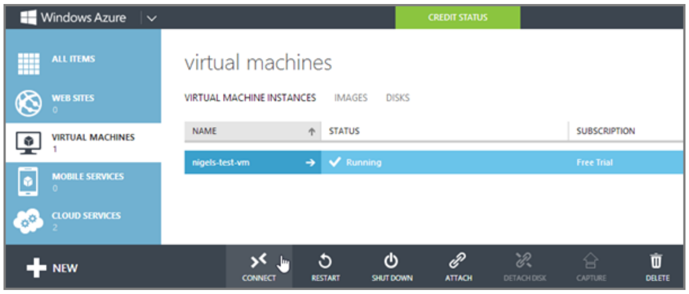
NAME	STATUS
nigels-test-vm	Starting (Provisioning)

1 OPERATION IS CURRENTLY RUNNING, 1 HAS COMPLETED

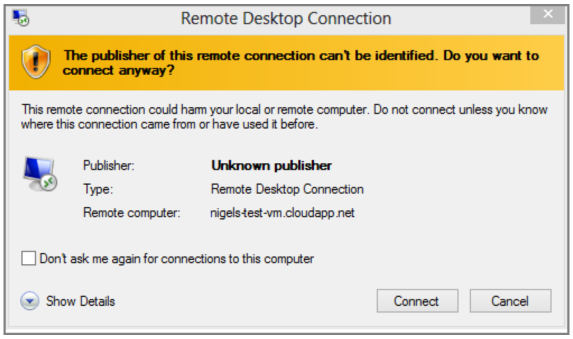
Creating the virtual machine...

Successfully deleted virtual machine nigels-neww-vm.

3. Now that your VM is created and running, you can log on. To log on to a Windows Azure VM, highlight the VM in the list of available VMs and click Connect.



The Windows Azure portal now downloads a Remote Desktop Protocol (RDP) app that will allow you to log on to your newly created VM. Opening the RDP app brings up the Remote Desktop Connection dialog box.


Remote Desktop Connection

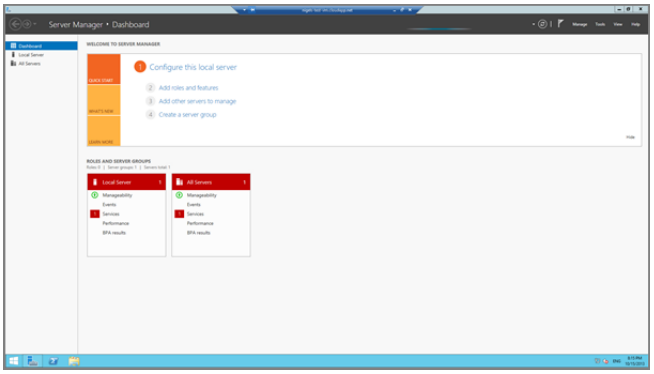
The publisher of this remote connection can't be identified. Do you want to connect anyway?

This remote connection could harm your local or remote computer. Do not connect unless you know where this connection came from or have used it before.

Publisher:	**Unknown publisher**
Type:	Remote Desktop Connection
Remote computer:	nigels-test-vm.cloudapp.net

☐ Don't ask me again for connections to this computer

4. Click the Connect button in the Remote Desktop Connection dialog box. Then enter your username and password in the Windows Security dialog box that appears. This logs you on to your new VM.

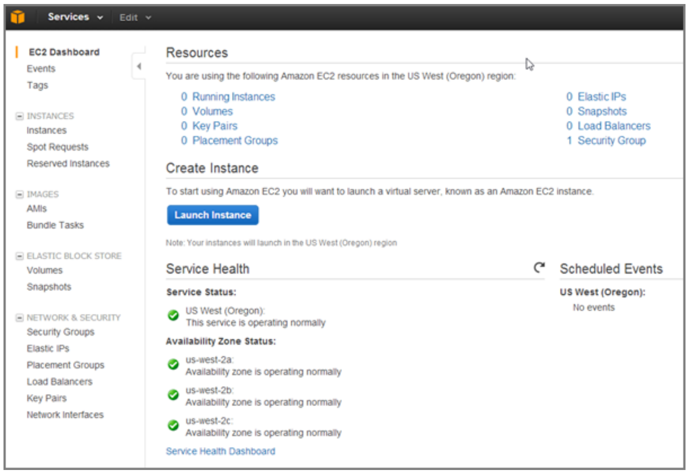

Server Manager • Dashboard
WELCOME TO SERVER MANAGER
Configure this local server
Add roles and features
Add other servers to manage
Create a server group
ROLES AND SERVER GROUPS
Local Server
All Servers

EXERCISE 15.2

Creating a New Virtual Machine in the Cloud with Amazon Web Services

In this exercise, you'll walk through the process of creating a new Linux VM in the EC2 component of AWS. This exercise assumes you have an Amazon Web Services account.

1. Once logged on to the AWS Management Console, the first thing to do is navigate to the EC2 Virtual Servers in the Cloud dashboard.
2. When in the EC2 dashboard, you need to create a new VM. In EC2, a VM is called an *instance*. In the EC2 dashboard web user interface, select the option Launch Instance.



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

0 Running Instances	0 Elastic IPs
0 Volumes	0 Snapshots
0 Key Pairs	0 Load Balancers
0 Placement Groups	1 Security Group

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

✓ US West (Oregon):
This service is operating normally

Availability Zone Status:

✓ us-west-2a:
Availability zone is operating normally

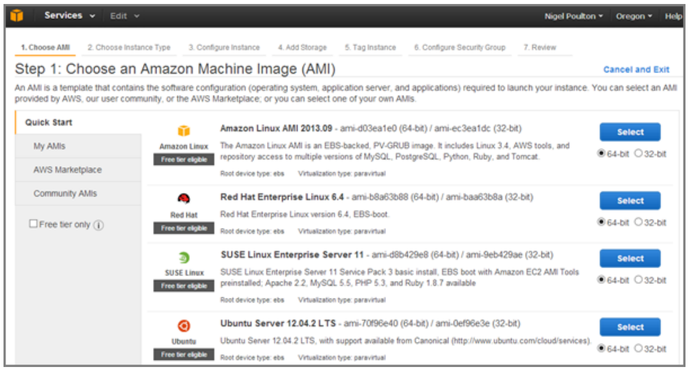
✓ us-west-2b:
Availability zone is operating normally

✓ us-west-2c:
Availability zone is operating normally

Scheduled Events

US West (Oregon):
No events

3. From the resulting screen, select the type of VM you would like to create. Options include most of the popular operating systems as well as choices of 32-bit or 64-bit. Some options are shown in the following screenshot. Select your instance type according to your operating system details.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

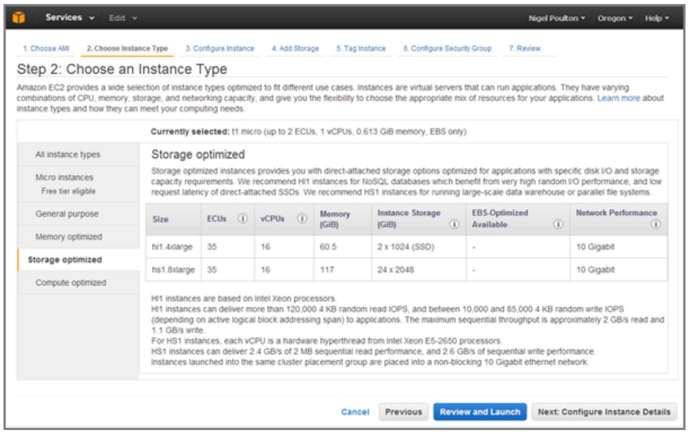
Amazon Linux AMI 2013.09 - ami-d03ea1e0 (64-bit) / ami-ec3ea1dc (32-bit)
The Amazon Linux AMI is an EBS-backed, PV-GRUB image. It includes Linux 3.4, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

Red Hat Enterprise Linux 6.4 - ami-b8a63b88 (64-bit) / ami-baa63b8a (32-bit)
Red Hat Enterprise Linux version 6.4, EBS-boot.
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

SUSE Linux Enterprise Server 11 - ami-d8b429e8 (64-bit) / ami-9eb429ae (32-bit)
SUSE Linux Enterprise Server 11 Service Pack 3 basic install, EBS boot with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

Ubuntu Server 12.04.2 LTS - ami-70f96e40 (64-bit) / ami-0ef96e3e (32-bit)
Ubuntu Server 12.04.2 LTS, with support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

4. You are now prompted for details related to the virtual machine. EC2 gives you several options that are optimized for certain characteristics, such as Memory Optimized, Compute Optimized, and Storage Optimized. Choose the type of VM you wish to create and click Next: Configure Instance Details. The following screenshot shows a Storage Optimized VM configuration.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t1.micro (up to 2 ECUs, 1 vCPUs, 0.613 GiB memory, EBS only)

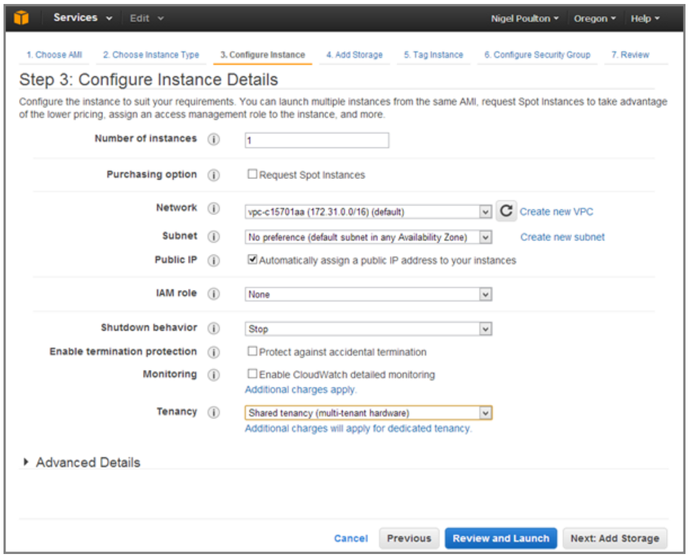
Storage optimized

Storage optimized instances provides you with direct-attached storage options optimized for applications with specific disk I/O and storage capacity requirements. We recommend HI1 instances for NoSQL databases which benefit from very high random I/O performance, and low request latency of direct-attached SSDs. We recommend HS1 instances for running large-scale data warehouse or parallel file systems.

Size	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
hi1.4xlarge	35	16	60.5	2 x 1024 (SSD)	-	10 Gigabit
hs1.8xlarge	35	16	117	24 x 2048	-	10 Gigabit

HI1 instances are based on Intel Xeon processors.
HI1 instances can deliver more than 120,000 4 KB random read IOPS, and between 10,000 and 85,000 4 KB random write IOPS (depending on active logical block addressing span) to applications. The maximum sequential throughput is approximately 2 GB/s read and 1.1 GB/s write.
For HS1 instances, each vCPU is a hardware hyperthread from Intel Xeon E5-2650 processors.
HS1 instances can deliver 2.4 GB/s of 2 MB sequential read performance, and 2.6 GB/s of sequential write performance.
Instances launched into the same cluster placement group are placed into a non-blocking 10 Gigabit ethernet network.

5. Click through the remaining screens to configure your new VM. This includes configuring the networking and shutdown details, some of which are shown here.

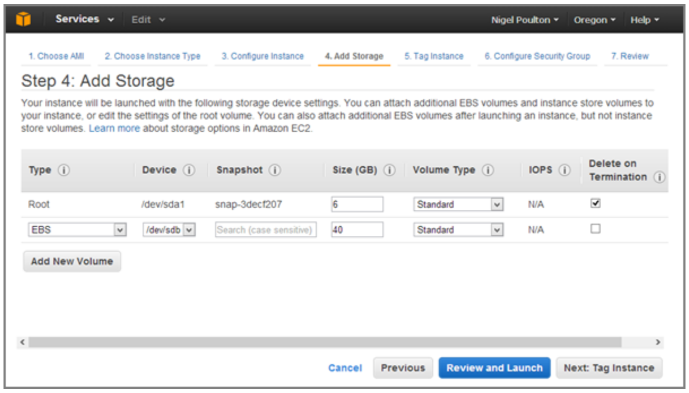


Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	vpc-c15701aa (172.31.0.0/16) (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zone) — Create new subnet
Public IP	☑ Automatically assign a public IP address to your instances
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared tenancy (multi-tenant hardware) Additional charges will apply for dedicated tenancy.

6. You can also add storage devices to your VM.



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Type	Device	Snapshot	Size (GB)	Volume Type	IOPS	Delete on Termination
Root	/dev/sda1	snap-3decf207	6	Standard	N/A	✓
EBS	/dev/sdb	Search (case sensitive)	40	Standard	N/A	☐

7. Once all of the information has been provided, click Review And Launch and your new VM will be created.

Private Cloud

The *private cloud* can have either of the following two designs:

- Your own internal cloud, with on-premise equipment that you still own and manage, but use to offer cloud-like services to your internal customers.
- Services hosted by a third party, off premise, but offering customers their own dedicated infrastructure (no multitenancy).

NOTE

The term *on premise* refers to hosting IT equipment in your own data center facilities (your own premises). The term *off premise* refers to hosting IT in somebody else's data center facilities. Sometimes the terms are shortened to *off prem* and *on prem,* respectively.

In the first model, where you own the on-premise equipment and provide cloud-like services to your internal customers, these services include things such as web portals that allow users to self-provision, as well as monthly itemized billing services. This is a common approach in large organizations, where the internal IT department is under pressure from the public cloud and needs to provide cloud-like services internally in order to discourage users from opting for public cloud services rather than services offered by the internal IT department.

In the second model, third parties provide customers with dedicated off-premise equipment. This provides some of the advantages of cloud services without some of the risks perceived around the public cloud. For example, an organization may have a desire to no longer own and manage all of its own technology infrastructure, but may also be averse to putting all of its data and services out to the public cloud on a massively shared platform. A private cloud option, in which another company takes care of all infrastructure management but provides a dedicated platform, may be a good solution. These options often allow customers to visit data center facilities and view the equipment that they are paying to use.

Hybrid Cloud

The *hybrid cloud*, as its name suggests, is a combination of the public cloud and private cloud. Many organizations aren't keen to throw all their data and services into the public cloud on day one. Instead, they prefer to start out by putting some of their less-important apps and services into the public cloud while keeping their mission-critical and sensitive apps in-house. This mixture of the public and private cloud is sometimes referred to as the *hybrid cloud*.

Hybrid clouds also allow customers to burst into the cloud as the business requires extra computing resources. For example, say your business requires a lot more computing capability at one particular time of the year—you put on an event once a year that sees a massive spike in visits to your website. Utilizing the public cloud allows you to increase your computing power once a year and then turn it off again (and stop paying for it) when it's no longer needed.

Storage and the Cloud

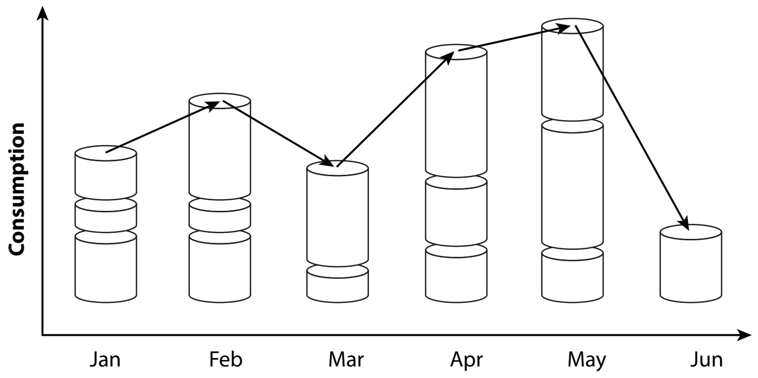
You've examined cloud computing. Now let's look more specifically at cloud storage.

NOTE *Storage as a service* is sometimes written as *SaaS*.

Cloud storage is accessed remotely over the Internet. You pay for it monthly using a consumption model where you only pay for what you're using.

The combination of elastic storage and consumption-based billing is shown in Figure 15.2.

FIGURE 15.2 Elasticity and consumption billing



Cloud storage has most of the features and benefits of cloud computing, including the following:

- Simplicity
- Elasticity
- Massive scalability
- Accessibility
- Fast self-service provisioning
- Built-in protection
- No cap-ex cost

Cloud storage doesn't suit every use case. Cloud storage services have some potential drawbacks that need considering. These include low performance when compared to non-cloud-based storage solutions. For example, accessing storage over an Internet connection is never going to be as fast as accessing storage over a short, low-latency Fibre Channel SAN that exists in close proximity to your servers within the confines of your own data center. Similarly, you won't want to put any latency-sensitive applications on public cloud storage. So that means no SQL-based relational databases on cloud storage.

Most cloud storage solutions are object storage–based. Although Chapter 7, "Files, NAS, and Objects," covers object storage (object stores) in detail, I'll remind you here that object storage is not designed for high-performance or structured data (databases), and it isn't ideal for data that changes a lot. However, it is good for streaming/sequential read workloads, making it ideal for storage and retrieval of rich media and other Web 2.0 types of content such as photo images, videos, audio, and other documents. Hence the reason that many of the popular social media sites use object storage systems to host their uploaded content such as images and videos, as these are uploaded once (written once), played back (read) multiple times, and almost never changed after being uploaded.

NOTE Make sure that you know whether your cloud provider charges you based on a base-2 or base-10 usage number. Base 10 thinks that 1 MB = 1,000 KB, whereas base 2 thinks that 1 MB = 1,024 KB. While it might seem a small difference with such small numbers, when you start talking about multiple terabytes (TB), the difference can be significant.

Data Durability

The concept of *data durability* is key to cloud storage. Data durability refers to the ability of an object stored in the cloud to survive failures within the cloud. You can think of this as being a little like RAID for the cloud.

NOTE The concept of data durability is borrowed from the database world, where the *D* in the well-known ACID acronym stands for *durability*. ACID, which stands for Atomicity, Consistency, Isolation, and Durability, outlines key features required to guarantee consistent and reliable database transactions.

At the time of this writing, Amazon S3 was claiming 99.999999999 percent durability. Amazon also states that this level of durability gives an average annual expected loss of objects of 0.000000001 percent. Basically, you'd be extremely unlucky to lose an object stored in S3, or as Amazon puts it, *If you store 10,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000,000 years*. Obviously, that is referring to the potential of losing objects due to failures within S3, and it doesn't cover you for user error such as accidentally deleting objects.

This kind of extreme data durability is achieved by cloud storage services storing multiple copies of each object throughout the cloud. For example, if you keep your data within the U.S. region of your chosen cloud, multiple copies of your data will be kept across multiple data centers within the U.S. region. This means that if a node, or even an entire data center, fails within the cloud's U.S. region, your data will still be available from other data centers within the same region.

Many cloud storage solutions also offer Geo replication, where objects are also replicated outside regions, across the globe. In a Geo replication scenario, your data is protected across multiple facilities within a region and then also across multiple regions. That is even more protection.

An important concept on the topic of data durability is *atomic updates*—remember, the *A* in ACID stands for *atomicity*! Atomic updates ensure that all updates to cloud storage (new objects or updates to existing objects) are stored to multiple locations within the cloud *before* the cloud confirms that the update was successful. So you know that once an object, or update to an object, is signaled as successfully committed to the cloud, that object has been stored to multiple locations.

NOTE

Despite the impressive numbers that cloud storage services advertise for data durability, you will still want to back up your data. Data durability on its own does not protect you against accidental deletions or unintentional updates to objects! If a cloud storage solution provides *versioning*—whereby the previous version of an object is preserved every time the object is updated or deleted—then this *may* allow you to stop backing up data that you have stored in the cloud. However, you will want to conduct thorough research before making a decision as important as changing your backup practices.

Eventual Consistency Model

Some aspects of cloud storage platforms work on an *eventual consistency* model. In this model, objects and updates to objects take time to propagate throughout the cloud. As you learned in the section about data durability, objects in a storage cloud are replicated, or propagated, to multiple nodes and locations within the cloud in order to protect against failures within the cloud. Sometimes this propagation of objects and updates to objects occurs asynchronously (such as propagating out of region) meaning that your updates to the cloud are signaled as complete before changes are replicated throughout the entire cloud. For example, it is possible in a cloud storage solution configured for Geo replication for an object to be updated in California in the U.S. region at 10 a.m., and for someone accessing the object from a copy in the APAC region at 10:01 a.m. (California time) to still see a previous version of the object, and for it to be 10:02 a.m. before the updated object is propagated to the APAC region. That being said, atomic updates are still respected, meaning that if your cloud storage solution is configured to make three copies *in region*, these three copies will be updated before any updates to your objects are signaled as complete; it is just the propagation out of the region that should be asynchronous and may take a while to propagate. As long as you leave the system for long enough, without making changes, all copies of your data in the cloud will eventually become consistent. Obviously, if rapid global consistency is important to your organization, cloud-based object storage may not be the best solution.

NOTE

Some examples of *eventual consistency* models—sometimes called *loose consistency*—include NoSQL databases and some social media and photo-sharing sites. In these solutions, it takes a short amount of time for changes and updates to be propagated throughout the system.

Public Cloud Storage

When most people think about cloud storage, they're thinking of *public cloud storage*, as it's the most popular form of cloud storage. Services such as Amazon S3, Windows Azure, and Rackspace Cloud Files are all examples of public cloud storage services.

At a high level, public cloud storage is storage that's accessed over the Internet, usually via RESTful application programming interfaces (APIs), and is paid for monthly based on what you've consumed that month. Public cloud storage is also multitenant by nature, meaning it has multiple customers sharing the same common infrastructure and storage platforms.

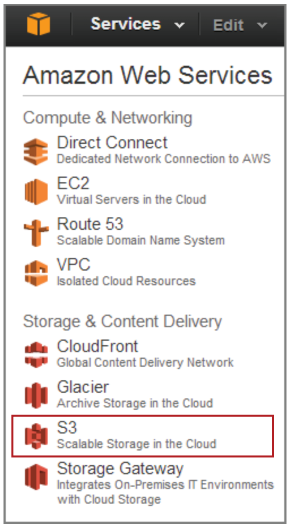
To help you understand how it works, Exercise 15.3 shows you how to configure some cloud storage in Amazon S3.

EXERCISE 15.3

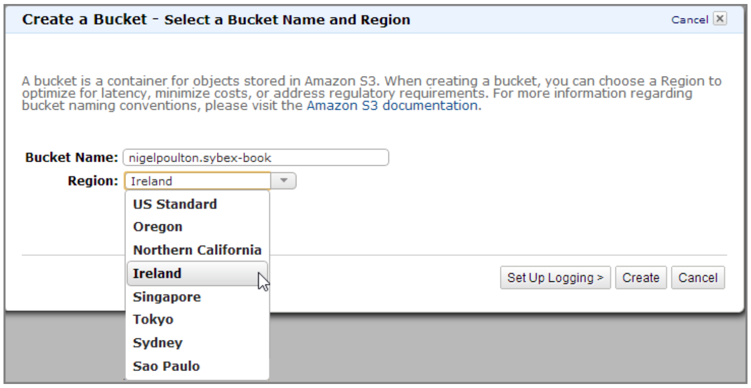
Getting Started with Public Cloud Storage with Amazon S3

In this exercise, you'll walk through the steps necessary to get up and running with Amazon Simple Storage Service (S3). Before you can get started with Amazon S3, you'll need an Amazon Web Services account. For this example, we'll assume you already have one. Now, you'll create a cloud storage bucket, upload an object, and configure custom metadata.

1. From within the AWS web console, navigate to the S3 service.


Services
Edit
Amazon Web Services
Compute & Networking
Direct Connect
Dedicated Network Connection to AWS
EC2
Virtual Servers in the Cloud
Route 53
Scalable Domain Name System
VPC
Isolated Cloud Resources
Storage & Content Delivery
CloudFront
Global Content Delivery Network
Glacier
Archive Storage in the Cloud
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage

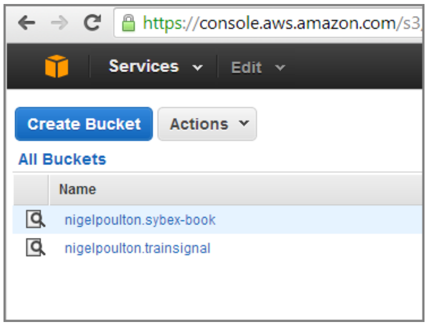
2. As all objects in Amazon S3 are stored in *buckets*, let's create a bucket. Click the Create Bucket button from within the S3 web interface.
3. In the resulting Create A Bucket screen, select a name for your bucket and choose a region that you would like your bucket stored in. In this example, I'm creating a bucket called nigelpoulton.sybex-book and choosing to store it in the Ireland region.



A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

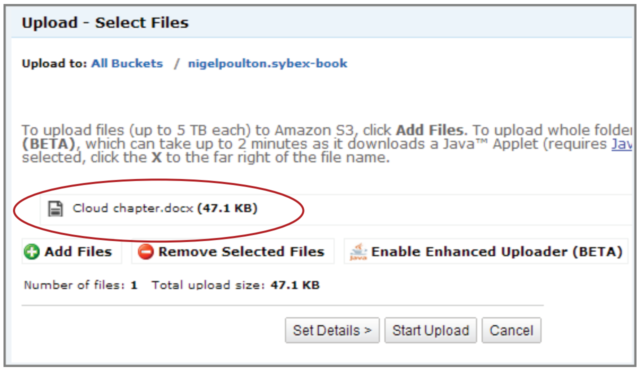
Bucket Name: nigelpoulton.sybex-book
Region: Ireland

4. Click Create. The newly created bucket appears in the list of buckets.


https://console.aws.amazon.com/s3

nigelpoulton.sybex-book
nigelpoulton.trainsignal

5. Now let's add an object (file) to the bucket. Choose your bucket and click Upload.
6. On the Upload screen, click Add Files and choose whatever files you want to upload to the bucket from the dialog box. As you can see in the following image, I have selected to upload the file for this chapter, currently called `Cloud chapter.docx`.



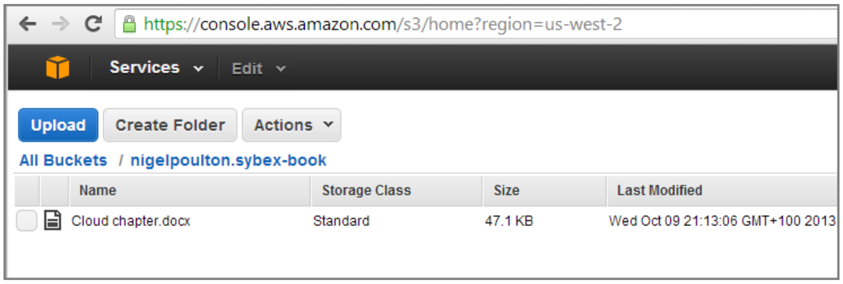
Upload to: All Buckets / nigelpoulton.sybex-book

To upload files (up to 5 TB each) to Amazon S3, click **Add Files**. To upload whole folde
(BETA), which can take up to 2 minutes as it downloads a Java™ Applet (requires Jav
selected, click the **X** to the far right of the file name.

Cloud chapter.docx **(47.1 KB)**

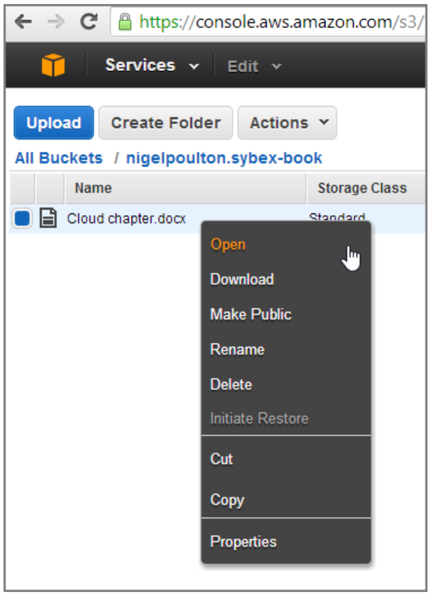
Number of files: **1** Total upload size: **47.1 KB**

7. Once your file is selected, click Start Upload. The following image shows that my `Cloud chapter.docx` object has been uploaded to the `mynigelpoulton.sybex-book` bucket.


https://console.aws.amazon.com/s3/home?region=us-west-2
Services
Edit
Upload
Create Folder
Actions
All Buckets / nigelpoulton.sybex-book

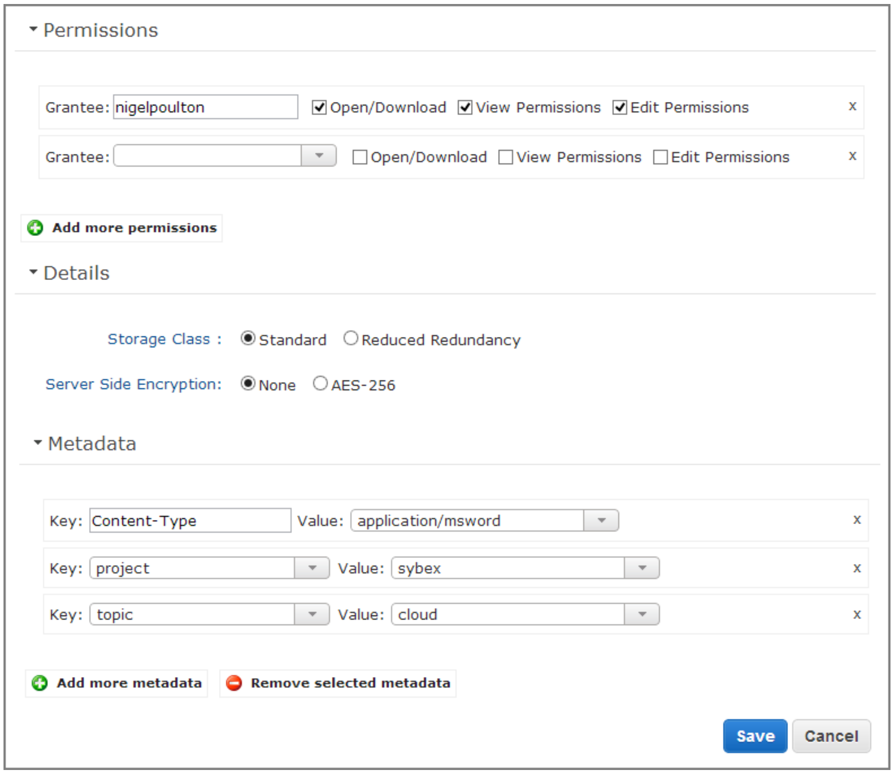
Name	Storage Class	Size	Last Modified
Cloud chapter.docx	Standard	47.1 KB	Wed Oct 09 21:13:06 GMT+100 2013

8. Right-click your uploaded object. You get a context menu that lists all the common options you can perform on the object.



9. From the right-click menu, select Properties. A page opens, enabling you to set common properties associated with objects stored in the Amazon S3 cloud, including these:

- Permissions
- Encryption settings
- Redundancy
- Custom metadata



Grantee: nigelpoulton [✓ Open/Download] [✓ View Permissions] [✓ Edit Permissions]

Grantee: Open/Download View Permissions Edit Permissions

Storage Class : [✓ Standard] Reduced Redundancy

Server Side Encryption: [✓ None] AES-256

Key:	Content-Type	Value:	application/msword
Key:	project	Value:	sybex
Key:	topic	Value:	cloud

And there you go. You've just created a cloud storage bucket, uploaded an object, and configured custom metadata.

API Access

As you can see from Exercise 15.3, Amazon S3 storage can be accessed via a web interface. Uploading objects, downloading objects, deleting objects, managing access to objects, and so on can all be done via a web browser. You don't need to mount cloud storage as you do with a SMB/CIFS share or an NFS export—in fact, you can't. And you don't need to format it with a filesystem as you do with a SCSI LUN. Again, you can't. Instead, you access it directly via either a web browser or application that talks directly to it using REST APIs.

As noted earlier, API is an acronym for *application programming interface*, and APIs are kind of like a published set of instructions for accessing and using web-based services. In the context of the public cloud, APIs are almost always RESTful. By RESTful, I mean a system that uses simple put/get style semantics such as the common HTTP `PUT` and `GET` verbs used so extensively on the World Wide Web. Most REST-based systems use HTTP (although they don't have to), and most, although not all, object storage systems are accessed via RESTful APIs.

If APIs are new to you, it might be useful to think of them as being similar to a set of rules and protocols you should follow in order to get something done. For example, the protocol for ordering food at a five-star restaurant is very different from ordering food at a McDonald's. In the five-star restaurant, you are respectfully ushered to your seat and given a menu, you then leisurely browse the menu and wait for the waiter to come over and take your order for drinks, and you then wait until the drinks are delivered before making your food order. You normally keep a tab open and settle the bill after you have eaten. Knowing that helps make your experience enjoyable. But try that same approach in a McDonald's restaurant and you will find yourself very hungry. There will be nobody waiting to seat you and no waiter coming to your table to take your order. And don't even bother trying to eat your food before you've paid for it! APIs aren't that much different. The Amazon S3 API gives you instructions to follow in order to access data in the S3 cloud. Follow the S3 API, and your experience with S3 will probably be good. Don't follow the S3 API, and your experience with S3 will not be so good.

APIs are also software-to-software interfaces, not user interfaces. They are designed to be used for programmatic or systematic access to a system, without requiring user assistance. On this topic, many APIs have *language bindings* to many of the popular programming languages such as Python, Java, PHP, C#, and so on. These language bindings are a higher level (simpler) than pure REST, allowing you to do RESTful operations within the native context of the programming language, making programming APIs a lot simpler.

Public Cloud Storage Performance

When it comes to performance, cloud storage is close to the bottom of the pile, which makes sense. Its lack of proximity to your on-premise systems or applications suggests that cloud storage won't be a stellar performer. Storage performance almost always gets slower as it gets further away from the CPU. For example, main memory (RAM) is close to the CPU and is extremely fast. Locally attached disk is a bit further away from the CPU (you access it over a PCI bus on the system motherboard) and is slower. Network-attached storage (SAN and NAS) is even further away and is slower again. And cloud storage is the furthest away of them all! So not surprisingly, it's the slowest of them all.

It's also interesting to note that the closer you get to the CPU, the less capacity is available, whereas the further away from the CPU, the more capacity. For example, RAM is closer to the CPU on a system than a hard drive is, and a hard drive is closer than a NAS array. From this we should know that RAM capacity is usually a lot less than hard drive capacity, which in turn is usually a lot less than the capacities provided by a NAS array. Cloud storage is true to that rule too, as it is the most scalable form of storage available today.

While we're talking about proximity to the CPU, it also appears to be true that the further away from the CPU your storage is, the more accessible and shareable it is. For example, sharing main memory (RAM) is relatively difficult compared to sharing space on a locally attached hard drive. Likewise, NAS arrays were designed with shareability in mind, as was cloud storage. Cloud storage provides the ultimate in storage accessibility and shareability.

With all of this said, cloud storage has its use cases. In some use cases, such as accessing large objects such as photos and video files, cloud storage can be a good performer. This is because access to these types of objects requires good throughput rather than low latency; a single HTTP `GET` can deliver a very large amount of data. And Internet bandwidth is getting cheaper and cheaper.

Atomic Uploads

All good public cloud storage solutions work based on the principle of *atomic uploads*. This is a good thing. Atomic uploads ensure that an object is not available to be accessed from the cloud until the upload operation is 100 percent complete. That atomic upload process usually involves making several copies of the object before signaling the upload as complete. This means that failures within the cloud should never result in you losing your objects. Atomic uploads include any associated metadata and security properties.

Geo Replication

Public cloud services often offer global replication capability, sometimes called *Geo replication*. This is where objects can be replicated across the cloud's worldwide infrastructure. This can obviously improve data durability—the ability of your data to survive large-scale local disasters and failures.

Another major use for Geo replication is local access to data. For example, if your public cloud provider has data centers in the United States, the United Kingdom, and Singapore, and you have offices in the United States, the United Kingdom, and Australia, you might want your objects replicating to all of your cloud provider's facilities so that staff in each of your locations around the world can access data without always having to come back to the United States. However, for legal or regulatory purposes, you may not be allowed to have your data leave a certain region or legal jurisdiction. For example, the European Union (EU) has laws that prohibit certain data from leaving the EU. Also, you may be concerned about government access to data that is stored within various countries.

Content Delivery Networks

On the topic of local access to data, the closer your data is to your users, the better the performance and the greater the chance of a good user experience. For example, if your public cloud provider has data centers in the United States, the United Kingdom, and Singapore, and you have users in Hong Kong, you'll probably want your users in Hong Kong to have access to data via the cloud provider's facilities in Singapore. However, it's still a long way from Hong Kong to Singapore. And this is where content delivery networks (CDNs) come to the rescue. CDNs work by caching objects closer to users. They are able to do this by having caching nodes in *edge locations* dotted around the globe. For example, Akamai Technologies, a CDN provider that partners with Rackspace, can cache objects in Hong Kong so users in Hong Kong have local access to data rather than having to traverse backward and forward across the Internet to Singapore every time they want access to an object.

NOTE Sometimes edge locations are referred to as *points of presence (POP)*.

So CDNs take the *access your data from any device, anywhere in the world* philosophy of cloud storage a step further, making it *access your data from any device, anywhere in the world, FAST!*

NOTE You may sometimes hear CDN explained as a content *distribution* network, rather than a content *delivery* network. Both terms mean the same thing. Some cloud solutions have their own CDN, such as Amazon S3, whereas others partner with a third-party CDN specialist such as Rackspace partnering with Akamai for their Cloud Files cloud storage platform. As an example, Akamai (a CDN provider) has over 100,000 servers in more than 200 *edge locations* dotted around the globe that can cache data so that it is closer to your users. On the cost side, you will also want to check whether there are additional charges for using a CDN.

While public cloud storage might pose challenges and risks to you and your organization, there is definitely a trend within organizations to offload their lower-tier requirements and their lower-risk data and services to someone else—and that someone else is increasingly becoming the public cloud.

Real World Scenario

Using Public Cloud Storage for Small Offices

Many organizations are starting to utilize public cloud storage offerings to facilitate storage requirements in remote branch offices where no IT staff are employed. A popular trend is pushing backups to the cloud, as volumes of data in small offices tend not to be large, and are therefore well suited to backup to the cloud. For example, rather than having physical tapes, or even a small dedicated disk unit/VTL at a remote branch office to cater for backup storage, many organizations are starting to send regional office backups to the cloud, via interfaces such as AWS VTL interface (Gateway-VTL). This avoids the need to deploy dedicated hardware at remote branch office, and removes any need for manual tape handling etc.

Private Cloud Storage

Private cloud storage comes in two major flavors:

- On premise
- Off premise

A common characteristic of both on premise and off premise is that customers of private cloud storage get their own private infrastructure—their own dedicated storage systems. This is in stark contrast to public cloud storage, where thousands of customers share the same underlying infrastructure and storage systems.

Despite being the polar opposite of public cloud storage when it comes to tenancy, private cloud storage services aim to emulate many of the other features of public cloud storage, including these:

- Simplicity
- Elasticity
- Scalability
- Fast self-service provisioning
- Built-in protection/data durability
- Op-ex-based consumption billing

In an *on-premise* private cloud storage solution, you access storage that is owned by a third party but resides in your data center, over your corporate LAN. In an *off-premise* private cloud storage solution, the storage is moved to the service provider's data center and is accessed over either the Internet or a private WAN link.

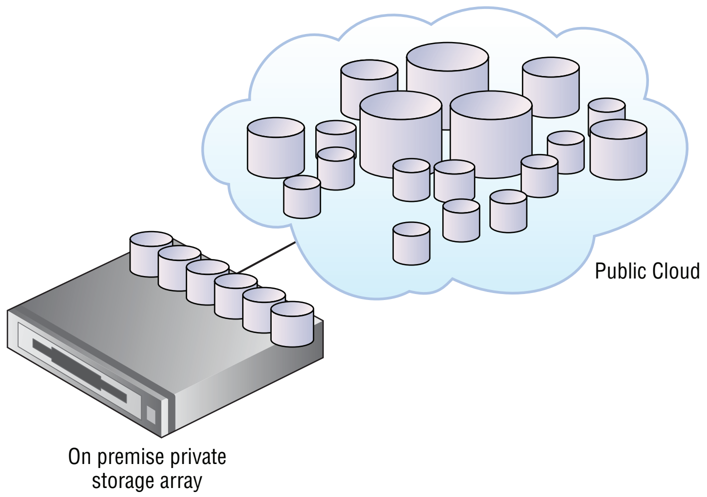
For companies that are nervous about letting their corporate data out of their hands and onto a massively shared public platform, private cloud storage may be a more palatable solution than public cloud storage. It may also be a good stepping-stone or halfway house, while companies warm up to the idea of public cloud storage.

Hybrid Cloud Storage

In *hybrid cloud storage*, a single application or product uses a mix of public cloud storage and private storage, giving the best of both worlds. Some examples of a commercially available hybrid storage solution include Microsoft StorSimple, Nasuni, Panzura, and TwinStrata. As an example, Microsoft StorSimple is a hardware and software storage array that you buy and locate in your data center but that can natively use the Windows Azure cloud as a low tier of storage. At a high level, the StorSimple product comprises an on-premise hardware storage array with internal SSD and SAS drives, as well as native integration with Windows Azure cloud storage. This can give the performance benefits of a traditional on-site storage system with SSD and SAS drives for fast access to frequently referenced data, as well as making Windows Azure appear as a tier of storage for storing less frequently accessed data. Microsoft is calling StorSimple *cloud integrated storage (CIS)*, as well as calling the use of Windows Azure *cloud as a tier (CaaT)*.

Figure 15.3 shows a simple hybrid cloud storage system such as Microsoft StorSimple.

FIGURE 15.3 Hybrid cloud storage device



Another common example of an increasingly common hybrid cloud storage solution is *hybrid cloud backup*. In a hybrid cloud backup solution, a staging appliance is located in your data center to store one to two weeks' worth of backups *locally* so they can be used to perform fast restores. Those backups are moved to the cloud after those one to two weeks.

CDMI

A standards-based API that you may come across in relation to cloud storage is *Cloud Data Management Initiative* (*CDMI*). This SNIA initiative aims to bring some standards to the world of cloud storage—standards for storing, accessing, and managing objects stored in the cloud. However, CDMI is practically irrelevant because of the massive popularity of other APIs including S3, OpenStack Swift (Cloud Files), and Azure. Most people who are coding for cloud storage are more commonly coding against S3, Swift, and Azure. Not as many people are coding against CDMI. However, more object storage systems are starting to support it.

De Facto and De Jure Standards

There are two major types of standards to be aware of:

- De jure standards
- De facto standards

De jure standards are those handed down to us by standards bodies and committees such as SNIA, IEEE, and ISO. *De facto standards* aren't controlled and handed down to us by standards bodies, but are so widely used in the real world as to be potentially more important than de jure standards. In the cloud storage space, CDMI is an example of a de jure standard that is written and regulated by a standards body, but not very widely adopted and practiced. Conversely, S3 and Swift are so widely used that they become almost unofficial standards that everybody abides by and practices.

Both have their pros and cons.

Architecturally speaking, CDMI-based solutions look and feel like any other cloud-based object store. CDMI stores objects in *containers* that are flat namespaces similar to buckets in Amazon S3. Objects are stored within containers, and containers cannot be nested. All objects within a CDMI cloud store are given a URL by which they can be accessed and manipulated via REST using the common HTTP verbs such as `PUT`, `GET`, `DELETE`, and so on. CDMI also supports other essential cloud storage features such as security and authentication.

CDMI is implemented as a client-server architecture. A CDMI client makes CDMI requests to a cloud storage service. The CDMI server in the cloud storage solution then represents data back to the CDMI client. This all occurs over HTTP/HTTPS and is very web browser and web friendly as the CDMI server is built on top of HTTP. An example of a client requesting to view a PDF file called `cloud-chapter.pdf` in a container called `sybex-book` on a cloud platform accessed at `http://cloud.provider.com` might look like the following:

```
GET /sybex-book/cloud-chapter.pdf HTTP/1.1
Host: cloud.provider.com
Accept: */*
```

As well as being able to create, retrieve, update, and delete data in the cloud, CDMI supports the ability to discover the capabilities of a particular cloud storage solution.

Summary

In this chapter, you learned the basics of the cloud computing model and the differences between the three popular forms of cloud computing: public cloud, private cloud, and hybrid cloud. You learned how the cloud is simple, elastic, scalable, self-service, and paid for on a pay-for-what-you-use model. You then learned about storage-specific cloud concepts including performance, atomic updates, data durability, and API access. Throughout the chapter, you touched on what the cloud may, and may not, be a good solution for. You also walked through examples of creating virtual servers and object stores in some of the most popular public cloud services available.

Chapter Essentials

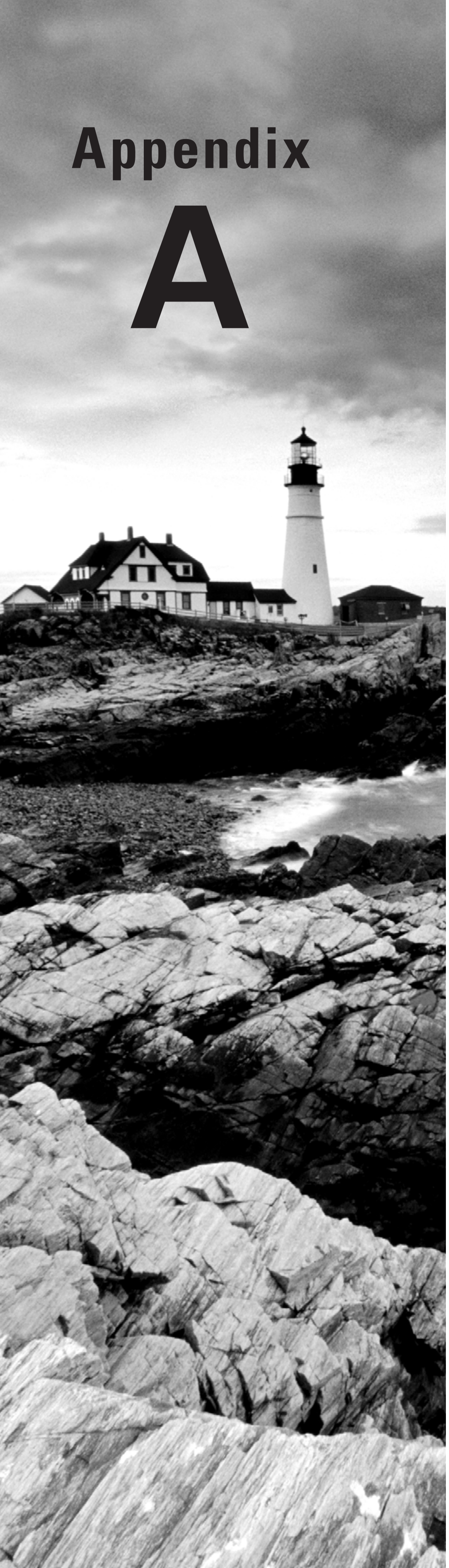
Cloud Computing In the cloud computing model, computing resources such as CPU, network, and storage are accessed remotely via the Internet. This model has several advantages over more-traditional computing models, such as elasticity, scalability, and accessibility.

Cloud Storage Cloud storage is accessed over the Internet via APIs. It shares many of the advantages of cloud computing, including elasticity, scalability, and accessibility.

Public Cloud Public cloud services, including cloud computing and cloud storage, operate on a shared infrastructure. Multiple customers sharing the same infrastructure is known as multitenancy.

API Access Application programming interfaces (APIs) allow services such as cloud computing and cloud storage to be accessed from within an application. As long as an API for a cloud service is made publically available, you can write your own code (software) to speak directly with that cloud service.

Appendix A



The CompTIA Storage+ Exam

TOPICS COVERED IN THIS APPENDIX:

✓ **Preparing for the CompTIA Storage+ exam**

✓ **Registering for the exam**

✓ **Taking the exam**

✓ **Reviewing exam objectives**

As mentioned in the introductory chapter to the book, IT is becoming more and more important in just about everything we do. In the corporate and business IT world, storage is a key IT system that requires skilled and trained professionals who can design and manage highly available and performant systems. One way to demonstrate your ability in this area is to attain the CompTIA Storage+ Powered by SNIA certification.

To achieve this certification, you need to pass the following exam:

CompTIA Storage+ Powered by SNIA Exam SGO-001 The Storage+ exam is the only storage certification offered by CompTIA in association with the Storage Networking Industry Association (SNIA). Covering all the topics required to configure basic storage networks and systems, this exam provides a solid foundation for building a more in-depth storage skill set. The exam is aimed at professionals who have at least one year of hands-on experience.

This book covers all the exam objectives, plus more, to enable you to pass the exam and be successful in your carreer as a storage and systems administrator.

Preparing for the Exam

Preparation is key to being successful. Now that you've read this book and are ready to pass the exam, it's a good idea to consider how the exam will look and feel, so that nothing takes you by surprise.

NOTE It's highly recommended to visit the CompTIA website before taking the exam, because details about the exam can change. The CompTIA website for this exam is `http://certification.comptia.org/getcertified/certifications/storage.aspx`.

The first thing to be aware of is that the exam is vendor neutral. There won't be any Dell Compellent or HP 3PAR questions on the exam. They're all more universal and conceptual. That is a good thing, as the information you've learned in this book is transferrable across most, if not all, vendors.

The exam is 1 hour and 30 minutes long and consists of 100 multiple-choice questions. There are no hands-on exercises or lab simulations. To pass, you need a score of 70% or more.

The exam is broken into the domains of expertise shown in Table A.1.

TABLE A.1 Exam Objective Categories

Expertise Domain	% of Exam
1.0 Storage Components	20%
2.0 Connectivity	24%
3.0 Storage Management	26%
4.0 Data Protection	17%
5.0 Storage Performance	13%

Looking at the objectives table, if you feel you're weak in any particular area, do some more homework so that you can go into the exam as confidently as possible.

Taking the Exam

At the time this book goes to press, the CompTIA Storage+ Powered by SNIA exam can be taken only at Pearson VUE testing centers. Pearson VUE is a leading global training company with more than 5,000 testing centers dotted around the globe.

To book your exam, you simply go to `www.pearsonvue.com/comptia` and follow the online procedure. The website enables you to locate your nearest testing center, book the test, and pay for it. To complete the procedure, you'll need a Pearson VUE account (which can be set up on the website), and an electronic payment method. The exam costs around $200 (or the local equivalent in whatever country you are in).

After your exam is booked, you need to sort out a few other important things, such as these:

- Plan your route to the testing center—you do not want to arrive late!
- Be sure to bring the required identification. You cannot take the exam without providing identification documentation that has a photograph of you and your current valid signature.

Once you know where the testing center is and you have the required identification, you're ready for the day of your exam.

On the day of your exam, do yourself a favor and arrive at the testing center early. The last thing you need is the stress of running late. Be prepared to surrender all personal electronic devices and have them stored in a secure locker while you take the exam.

After signing in at the testing center, you'll be taken to a secure room, where you'll be logged in to a computer and asked to complete a short computer-based survey. The time required to take this survey has no effect on the time you have to take the exam, so take your time with the survey. The answers you give in the survey do not influence your exam marks.

After you've completed the survey, it's time to take the exam. As mentioned earlier, the exam is multiple-choice questions. You have the ability to mark questions for later review: if you're unsure of the answer to a question, you can mark that question for review, and after you've answered all other questions, the system will allow you to go back to any questions that you've marked. Do your best to answer all questions! There is no advantage in leaving a question unanswered. An unanswered question is automatically considered incorrect. If you guess at the answer, you stand a chance of guessing right. So answer all questions!

After you've answered all questions and clicked the Complete button, you'll be shown your score and whether you've passed the exam. Hopefully, you've passed, and the system will automatically print out your score card. You'll then receive an email outlining your results and giving you the option of having a certificate mailed to you. In the event that you do not pass the exam, you are able to retake the exam at any time convenient to yourself and your chosen testing center—there is no requirement for a waiting period between your first and second attempt at the exam. However, if you do not pass on your second attempt, you will need to wait a minimum of 14 days between any further retakes.

Reviewing Exam Objectives

Table A.2 provides objective mappings for the SGO-001 exam. You can also find coverage of exam objectives in the flash cards and practice exams on the book's companion website, `http://www.sybex.com/go/datastoragenetworking`.

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map

1.0 Storage Components	Corresponding Book Chapter
1.1 Describe disk types, components, and features. • SATA • Fibre Channel • SAS • SCSI • SSD • Spindle • Platter • Cylinder • Heads • Speeds o 7,200 rpm o 10,000 rpm o 15,000 rpm • I/O vs. throughput • Capacity vs. speed	All covered in Chapter 2
1.2 Compare removable media types, components, and features. • Tape o Size vs. speed o Multistreaming and multiplexing (pros and cons) o Shoe-shining o LTO versions (LTO1, LTO2, LTO3, LTO4, LTO5) o Compression and encryption (hardware/software) o NDMP • Other removable media o DVD o Blu-ray Disc o Flash drives o WORM	Chapter 2 Chapter 10 Chapter 11

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

1.3 Given a scenario, install and maintain connectors and cable types (keeping in mind their properties). • Fiber cables o Multimode (shortwave) vs. single-mode (long-wave) o Length, speed, and distance limitations o Connectors: LC, SC, SFP o Care of cables: bend radius, stress • Copper cables o CAT 5 o CAT 5e o CAT 6 o Serial o Twinax o SAS o Connectors: RJ-45, BD-9 • SAS1 and SAS2 port speeds	Chapter 13 Chapter 14 Chapter 5
1.4 Describe the uses of physical networking hardware. • Switch and features o Trunking o ISL o Port channel o Port types: G_Ports, F_Ports, N_Ports, E_Ports, U_Ports o Directors o Hot-pluggable • HBA • CNA • Router	Chapter 5 Chapter 6

1.5 Given a scenario, install and maintain modular storage array components. • Controller head o Single o Dual o Grid o Cache o Expansion adapters o Array port types and cabling: Fibre Channel, FCoE, iSCSI, SAS • Disk enclosure o Enclosure controllers o Monitoring cards o Enclosure addressing o Cabling • Hot-pluggable	Chapter 3 Chapter 5
1.6 Identify the following environmental concerns and their associated impacts. • HVAC o Improper cooling o Adequate humidity control • Fire suppression • Floor and rack loading • Adequate power o Sufficient capacity o Adequate division of circuits o Grounding	Chapter 13
1.7 Use appropriate safety techniques during installation and maintenance of storage equipment. • Proper lifting techniques • Weight considerations • Antistatic devices • Rack stabilization	Chapter 13

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

2.0 Connectivity	
2.1 Identify common storage networking industry terms. • Link • Oversubscription • Worldwide node name • Worldwide port name • Flow control • N_Port ID • Buffer-to-buffer credit	Chapter 5
2.2 Explain the following storage networking industry terms. • Alias • Name service • Link • Connection • Initiator • Target • Fabric	

2.3 Given a scenario, implement the following Fibre Channel technologies. — Chapter 5

- Topologies
    - o Point-to-point
    - o Arbitrated loop
    - o Single fabrics
    - o Redundant fabrics
- Implementations
    - o Zoning best practices
    - o Zoning alias
    - o Zone
    - o Zone set
    - o Hard zoning
    - o Soft zoning
    - o Domain IDs
    - o NPIV
    - o SCSI IDs
- Multipathing
    - o Load balancing
    - o Failover
    - o Number of paths to disks
    - o Interoperability
- Physical connections vs. logical connections
- Protocols
    - o SCSI
    - o FCP
    - o FCIP

2.4 Given a scenario, implement the following Ethernet network technologies. — Chapter 5

- Features
    - o VLAN
    - o WAN
    - o MAN
    - o LAN
- Multipathing
    - o iSCSI
    - o MPIO
    - o Link aggregation
- Protocols
    - o iSCSI
    - o NFS
    - o CIFS

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

2.5 Identify the basics of converged storage network technologies. • FCoE • DCB (DCE, CEE) • LLDP • Class of service • Priority tagging • Baby jumbo frames • 10G Ethernet	Chapter 5 Chapter 3 Chapter 6 Chapter 7
2.6 Given a scenario, use the appropriate network tools. • TCP/IP network o ping o tracert/traceroute o ipconfig/ifconfig o nslookup • Fibre Channel network o Port error counters o fcping o Name server o Rescan	Chapter 14
2.7 Troubleshoot the following common networking problems. • Bad cables • Bad ports • Bad connectors • Incorrect configuration on NIC • Incorrect VLAN • Bad NIC • NIC improperly connected • Incorrect firewall settings	Chapter 5 Chapter 6
2.8 Troubleshoot the following common Fibre Channel problems. • Zoning errors • Zoning misconfiguration • Failed GBIC or SFP • Failed HBA • Connectivity • Interoperability issues • Hardware/software incompatibility • Outdated firmware/drivers • Failed cable • Misconfigured Fibre Channel cable	Chapter 13 Chapter 5 Chapter 6

2.9 Compare and contrast common storage infrastructures.

- SAN
  - o Fibre Channel
  - o Block mode
  - o Filesystem on host
  - o FC or iSCSI protocol
  - o Fabric
- NAS
  - o TCP/IP based
  - o UNC addressable storage
  - o Filesystem on storage (NFS or CIFS)
  - o Ethernet based
- DAS
  - o Direct-attached storage (SAS, SATA, SCSI)
  - o Filesystem on host

Chapter 13
Chapter 5
Chapter 6

3.0 Storage Management

3.1 Explain the following RAID levels and associated properties.

- Levels
  - o 0
  - o 1
  - o 5
  - o 6
  - o 1+0 (10)
  - o 0+1
- Properties
  - o High read
  - o High write
  - o Fault tolerance
  - o Rebuild times
  - o Performance
  - o Failure modes
  - o Capacity overhead

Chapter 3
Chapter 5
Chapter 6
Chapter 7

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

3.2 Given a scenario, execute storage provisioning techniques.	Chapter 4
• LUN provisioning	
o LUN ID	
• LUN masking and sharing	
o Host-based vs. storage-based (disk/tape)	
o Load balancing	
• Thin provisioning	
o Thin reclamation	
• Best practices for disk provisioning	
3.3 Explain volume management concepts.	Chapter 3
• File- vs. block-level architecture	Chapter 10
• Configuration layer	
o LVM	
• Logical volume	
• Volume group	
• Filesystem	
• Mount point	
3.4 Describe general virtualization concepts.	Chapter 4
• Virtual storage	Chapter 3
o Tapes	Chapter 7
o Disk	
• Virtual provisioning of the host, array, and fabric	
• LVM	
• VSAN/virtual fabric	
• VLAN	
• NPIV	

3.5 Given a scenario, implement monitoring, alerting, and reporting. • Setting thresholds • Trending • Forecasting/capacity planning • Recording baseline • Setting alerts • Auditing log files • Alerting methods o Cell phone o Email o SNMP o Call home	Chapter 9 Chapter 11 Chapter 5 Chapter 6
3.6 Explain management protocols, interfaces, and associated purposes. • Management protocols o SNMP o SMI-S o WBEM • Administration o CLI o Serial o Telnet o SSH o HTTP/S • In-band vs. out-of-band management	Chapter 10 Chapter 12
3.7 Explain Information Lifecycle Management concepts. • Data migration strategies (HSM) and storage tiers • Archiving and purging • Compliance and preservation • Content-addressable storage (CAS)/object-oriented storage (OOS) • Value of data based on frequency of access	Chapter 12
3.8 Explain the various functions and differences of deduplication and compression. • Inline and post-process deduplication • Software based vs. appliance based • Single-instance storage • Performance and capacity implications • Reduction ratios vs. data type	Chapter 7 Chapter 3 Chapter 10 Chapter 11

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

4.0 Data Protection	

4.1 Explain redundancy concepts, associated purposes, and components.
• High availability
• Single point of failure
• Component redundancy
  o Power supply
  o Controller
  o Disks (hot spare)
  o Path/bus
  o Switches
  o HBA
  o NICs
  o Array
• Cache battery backup and cache mirroring
	Chapter 10

4.2 Compare and contrast different replication methods and properties. • Synchronous and asynchronous • Local vs. remote • Site redundancy • Snapshots and clones • Replication consistency	Chapter 3 Chapter 4 Chapter 5

4.3 Explain the basics of data backup concepts for long-term storage. Chapter 8

- Recovery point objective (RPO) and recovery time objective (RTO)
- Backup and restore methods
    - o Full
    - o Incremental
    - o Differential
    - o Progressive
- Backup implementation methods
    - o LAN-free
    - o Serverless
    - o Server-based
- Backup targets
    - o Disk-to-disk
    - o Disk-to-tape
    - o VTL
    - o D2D2T
- Vaulting vs. e-vaulting
- Verify backups
    - o Data integrity
    - o Checksums
    - o Application verification
- Data retention and preservation policy
    - o Rotation schemes (GFS—Grandfather, Father, Son)
    - o Corporate and legal compliance
    - o Off-site tape storage/disaster recovery plan

TABLE A.2 Exam SG0-001 CompTIA Storage+ Powered by SNIA Objectives Map *(continued)*

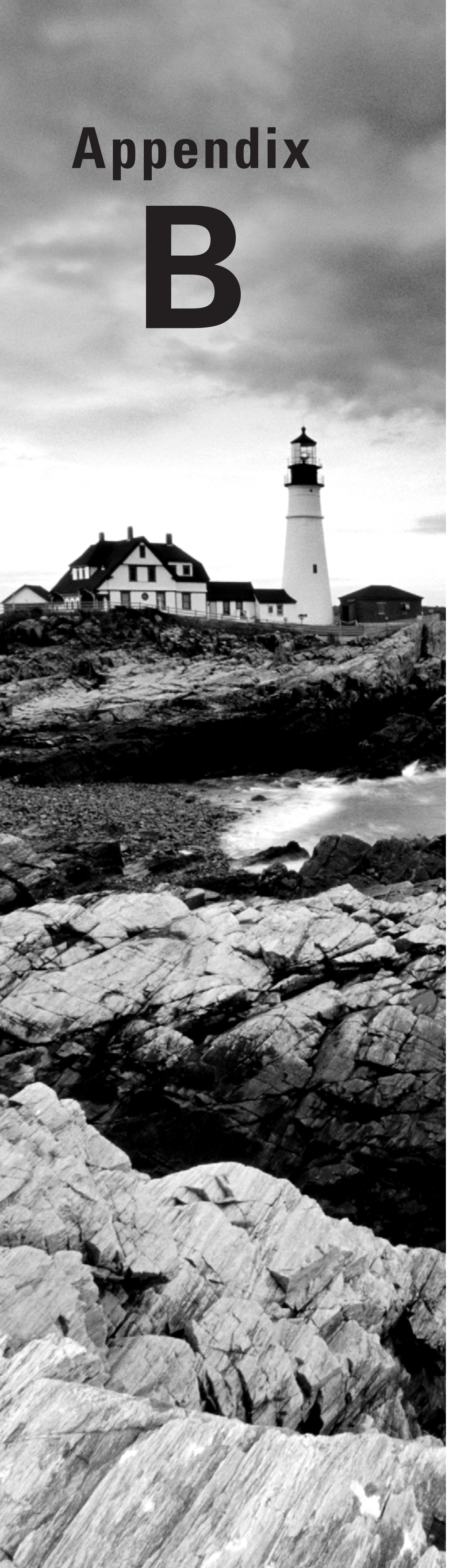
4.4 Explain the basic concepts and importance of data security.	Chapter 11
• Access management	
o ACL	
o Physical access	
o Multiprotocol/interoperability	
• Encryption	
o Disk encryption	
o Tape encryption	
o Network encryption (IPsec)	
o Host encryption	
o Encryption keys	
• Storage security	
o Shared access (NFSv3 vs. NFSv4)	
o Shared access (CIFS)	
o File permissions vs. share/export permissions	
5.0 Storage Performance	
5.1 Explain how latency and throughput impact storage performance.	Chapter 3 Chapter 11 Chapter 7
• Cache	
o Read vs. write traffic	
o De-staging	
o Cache hit and miss	
• RAID type and size	
o Number of disks	
• IOPS calculations	
• Random vs. sequential I/O	
• Impact of replication	
5.2 Identify tuning and workload balance concepts.	Chapter 3 Chapter 4 Chapter 8
• Application to storage data profiling	
• Tiering	
o Automatic	
o Manual	
o HSM	
• Partition alignment	
o Fragmentation and impact to performance	
• Queue depth	

5.3 Describe storage device bandwidth properties and functions.	Chapter 3
	Chapter 10
• Bus bandwidth/loop bandwidth	
• Cable speeds	
• Disk throughput vs. bus bandwidth vs. cache	
• Embedded switch port speed	
• Shared vs. dedicated	
• Multipathing for load balancing	
5.4 Describe network device bandwidth properties and functions.	Chapter 3
	Chapter 5
• Shared vs. dedicated	
• Teaming/link aggregation	
• Class of service	
o Jumbo frames	
• TOE	
5.5 Explain performance metrics, parameters, and purposes of storage/host tools.	Chapter 5
	Chapter 6
• Baselining and data capture	
• Switch	
o Port stats	
o Thresholds	
o Hops	
o Port groups	
o ISL/trunk	
o Bandwidth	
• Array	
o Cache hit rate	
o CPU load	
o Port stats	
o Bandwidth	
o Throughput	
o I/O latency	
• Host tools	
o sysmon	
o perfmon	
o iostat	

NOTE

Exam objectives are subject to change at any time without prior notice. Make sure you visit `http://certification.comptia.org/getcertified/certifications/storage.aspx`. If the URL for the exam has moved, Google probably knows where it is!

Good luck with the exam!



About the Additional Study Tools

In this appendix:

- ✓ **Additional study tools**
- ✓ **System requirements**
- ✓ **Using the study tools**
- ✓ **Troubleshooting**

Additional Study Tools

The following sections are arranged by category and summarize the software and other goodies you'll find from the companion website. If you need help with installing the items, refer to the installation instructions in the "Using the Study Tools" section of this appendix.

NOTE The additional study tools can be found at `www.sybex.com/go/datastoragenetworking`. Here, you will get instructions on how to download the files to your hard drive.

Sybex Test Engine

The files contain the Sybex test engine, which includes one bonus practice exam for the CompTIA Storage+ exam.

Electronic Flashcards

These handy electronic flashcards are just what they sound like. One side contains a question, and the other side shows the answer.

PDF of Glossary of Terms

We have included an electronic version of the Glossary in `.pdf` format. You can view the electronic version of the Glossary with Adobe Reader.

Adobe Reader

We've also included a link to download a copy of Adobe Reader so you can view PDF files that accompany the book's content. For more information on Adobe Reader or to check for a newer version, visit Adobe's website at `www.adobe.com/products/reader/`.

System Requirements

Make sure your computer meets the minimum system requirements shown in the following list. If your computer doesn't match up to most of these requirements, you may have problems using the software and files. For the latest and greatest information, please refer to the ReadMe file located in the downloads.

- A PC running Microsoft Windows 98, Windows 2000, Windows NT4 (with SP4 or later), Windows Me, Windows XP, Windows Vista, or Windows 7
- An Internet connection

Using the Study Tools

To install the items, follow these steps:

1. Download the `.ZIP` file to your hard drive, and unzip to an appropriate location. Instructions on where to download this file can be found here: `www.sybex.com/go/datastoragenetworking`.
2. Click the `Start.EXE` file to open up the study tools file.
3. Read the license agreement, and then click the Accept button if you want to use the study tools.

The main interface appears. The interface allows you to access the content with just one or two clicks.

Troubleshooting

Wiley has attempted to provide programs that work on most computers with the minimum system requirements. Alas, your computer may differ, and some programs may not work properly for some reason.

The two likeliest problems are that you don't have enough memory (RAM) for the programs you want to use or you have other programs running that are affecting installation or running of a program. If you get an error message such as "Not enough memory" or "Setup cannot continue," try one or more of the following suggestions and then try using the software again:

Turn off any antivirus software running on your computer. Installation programs sometimes mimic virus activity and may make your computer incorrectly believe that it's being infected by a virus.

Close all running programs. The more programs you have running, the less memory is available to other programs. Installation programs typically update files and programs; so if you keep other programs running, installation may not work properly.

Have your local computer store add more RAM to your computer. This is, admittedly, a drastic and somewhat expensive step. However, adding more memory can really help the speed of your computer and allow more programs to run at the same time.

Customer Care

If you have trouble with the book's companion study tools, please call the Wiley Product Technical Support phone number at (800) 762-2974, extension 74, or email them at `http://sybex.custhelp.com/`.

Index

Note to the Reader: Throughout this index **boldfaced** page numbers indicate primary discussions of a topic. *Italicized* page numbers indicate illustrations.