Unit 1 ISM - n/a

Information Storage And Management (SRM Institute of Science and Technology)

Studocu is not sponsored or endorsed by any college or university
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Chapter 1
Introduction to Information Storage

KEY CONCEPTS

- Data and Information
- Structured and Unstructured Data
- Evolution of Storage Architecture
- Core Elements of a Data Center
- Virtualization and Cloud Computing

Information is increasingly important in our daily lives. We have become information-dependent in the 21st century, living in an on-command, on-demand world, which means, we need information when and where it is required. We access the Internet every day to perform searches, participate in social networking, send and receive e-mails, share pictures and videos, and use scores of other applications. Equipped with a growing number of content-generating devices, more information is created by individuals than by organizations (including business, governments, non-profits and so on). Information created by individuals gains value when shared with others. When created, information resides locally on devices, such as cell phones, smartphones, tablets, cameras, and laptops. To be shared, this information needs to be uploaded to central data repositories (data centers) via networks. Although the majority of information is created by individuals, it is stored and managed by a relatively small number of organizations.

The importance, dependency, and volume of information for the business world also continue to grow at astounding rates. Businesses depend on fast and reliable access to information critical to their success. Examples of business processes or systems that rely on digital information include airline reservations, telecommunications billing, Internet commerce, electronic banking, credit card transaction processing, capital/stock trading, health care claims processing, life science research, and so on. The increasing dependence of businesses on information has amplified the challenges in storing, protecting, and managing

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

data. Legal, regulatory, and contractual obligations regarding the availability and protection of data further add to these challenges.

Organizations usually maintain one or more data centers to store and manage information. A *data center* is a facility that contains information storage and other physical information technology (IT) resources for computing, networking, and storing information. In traditional data centers, the storage resources are typically dedicated for each of the business units or applications. The proliferation of new applications and increasing data growth have resulted in islands of discrete information storage infrastructures in these data centers. This leads to complex information management and underutilization of storage resources. Virtualization optimizes resource utilization and eases resource management. Organizations incorporate virtualization in their data centers to transform them into *virtualized data centers* (VDCs). Cloud computing, which represents a fundamental shift in how IT is built, managed, and provided, further reduces information storage and management complexity and IT resource provisioning time. Cloud computing brings in a fully automated request-fulfillment process that enables users to rapidly obtain storage and other IT resources on demand. Through cloud computing, an organization can rapidly deploy applications where the underlying storage capability can scale-up and scale-down, based on the business requirements.

This chapter describes the evolution of information storage architecture from a server-centric model to an information-centric model. It also provides an overview of virtualization and cloud computing.

1.1 Information Storage

Organizations process data to derive the information required for their day-to-day operations. Storage is a repository that enables users to persistently store and retrieve this digital data.

1.1.1 Data

Data is a collection of raw facts from which conclusions might be drawn. Handwritten letters, a printed book, a family photograph, printed and duly signed copies of mortgage papers, a bank's ledgers, and an airline ticket are all examples that contain data.

Before the advent of computers, the methods adopted for data creation and sharing were limited to fewer forms, such as paper and film. Today, the same data can be converted into more convenient forms, such as an e-mail message, an e-book, a digital image, or a digital movie. This data can be generated using a computer and stored as strings of binary numbers (0s and 1s), as shown in Figure 1-1. Data in this form is called *digital data* and is accessible by the user only after a computer processes it.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

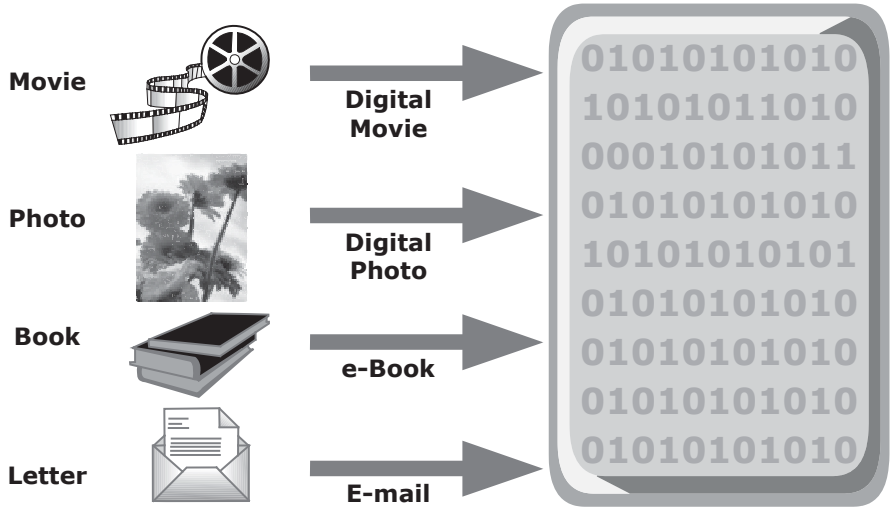


Figure 1-1: Digital data

With the advancement of computer and communication technologies, the rate of data generation and sharing has increased exponentially. The following is a list of some of the factors that have contributed to the growth of digital data:

- **Increase in data-processing capabilities:** Modern computers provide a significant increase in processing and storage capabilities. This enables the conversion of various types of content and media from conventional forms to digital formats.
- **Lower cost of digital storage:** Technological advances and the decrease in the cost of storage devices have provided low-cost storage solutions. This cost benefit has increased the rate at which digital data is generated and stored.
- **Affordable and faster communication technology:** The rate of sharing digital data is now much faster than traditional approaches. A handwritten letter might take a week to reach its destination, whereas it typically takes only a few seconds for an e-mail message to reach its recipient.
- **Proliferation of applications and smart devices:** Smartphones, tablets, and newer digital devices, along with smart applications, have significantly contributed to the generation of digital content.

Inexpensive and easier ways to create, collect, and store all types of data, coupled with increasing individual and business needs, have led to accelerated data growth, popularly termed *data explosion*. Both individuals and businesses have contributed in varied proportions to this data explosion.

The importance and value of data vary with time. Most of the data created holds significance for a short term but becomes less valuable over time. This governs the type of data storage solutions used. Typically, recent data which

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

has higher usage is stored on faster and more expensive storage. As it ages, it may be moved to slower, less expensive but reliable storage.

EXAMPLES OF RESEARCH AND BUSINESS DATA

Following are some examples of research and business data:

- **Customer data: Data related to a company's customers, such as order details, shipping addresses, and purchase history.**
- **Product data: Includes data related to various aspects of a product, such as inventory, description, pricing, availability, and sales.**
- **Medical data: Data related to the healthcare industry, such as patient history, radiological images, details of medication and other treatment, and insurance information.**
- **Seismic data: Seismology is a scientific study of earthquakes. It involves collecting data and processes to derive information that helps determine the location and magnitude of earthquakes.**

Businesses generate vast amounts of data and then extract meaningful information from this data to derive economic benefits. Therefore, businesses need to maintain data and ensure its availability over a longer period. Furthermore, the data can vary in criticality and might require special handling. For example, legal and regulatory requirements mandate that banks maintain account information for their customers accurately and securely. Some businesses handle data for millions of customers and ensure the security and integrity of data over a long period of time. This requires high-performance and high-capacity storage devices with enhanced security and compliance that can retain data for a long period.

1.1.2 Types of Data

Data can be classified as structured or unstructured (see Figure 1-2) based on how it is stored and managed. Structured data is organized in rows and columns in a rigidly defined format so that applications can retrieve and process it efficiently. Structured data is typically stored using a database management system (DBMS).

Data is unstructured if its elements cannot be stored in rows and columns, which makes it difficult to query and retrieve by applications. For example, customer contacts that are stored in various forms such as sticky notes, e-mail messages, business cards, or even digital format files, such as .doc, .txt, and .pdf. Due to its unstructured nature, it is difficult to retrieve this data using a traditional customer relationship management application. A vast majority of new data being created

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

today is unstructured. The industry is challenged with with new architectures, technologies, techniques, and skills to store, manage, analyze, and derive value from unstructured data from numerous sources.

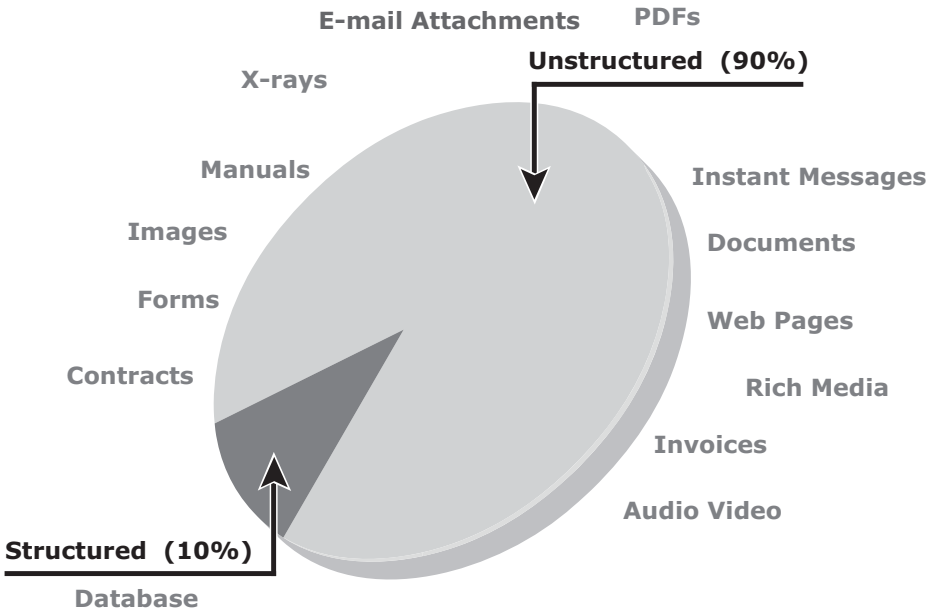


Figure 1-2: Types of data

1.1.3 Big Data

Big data is a new and evolving concept, which refers to data sets whose sizes are beyond the capability of commonly used software tools to capture, store, manage, and process within acceptable time limits. It includes both structured and unstructured data generated by a variety of sources, including business application transactions, web pages, videos, images, e-mails, social media, and so on. These data sets typically require real-time capture or updates for analysis, predictive modeling, and decision making.

Significant opportunities exist to extract value from big data. The big data ecosystem (see Figure 1-3) consists of the following:

1. Devices that collect data from multiple locations and also generate new data about this data (metadata).
2. Data collectors who gather data from devices and users.
3. Data aggregators that compile the collected data to extract meaningful information.
4. Data users and buyers who benefit from the information collected and aggregated by others in the data value chain.

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

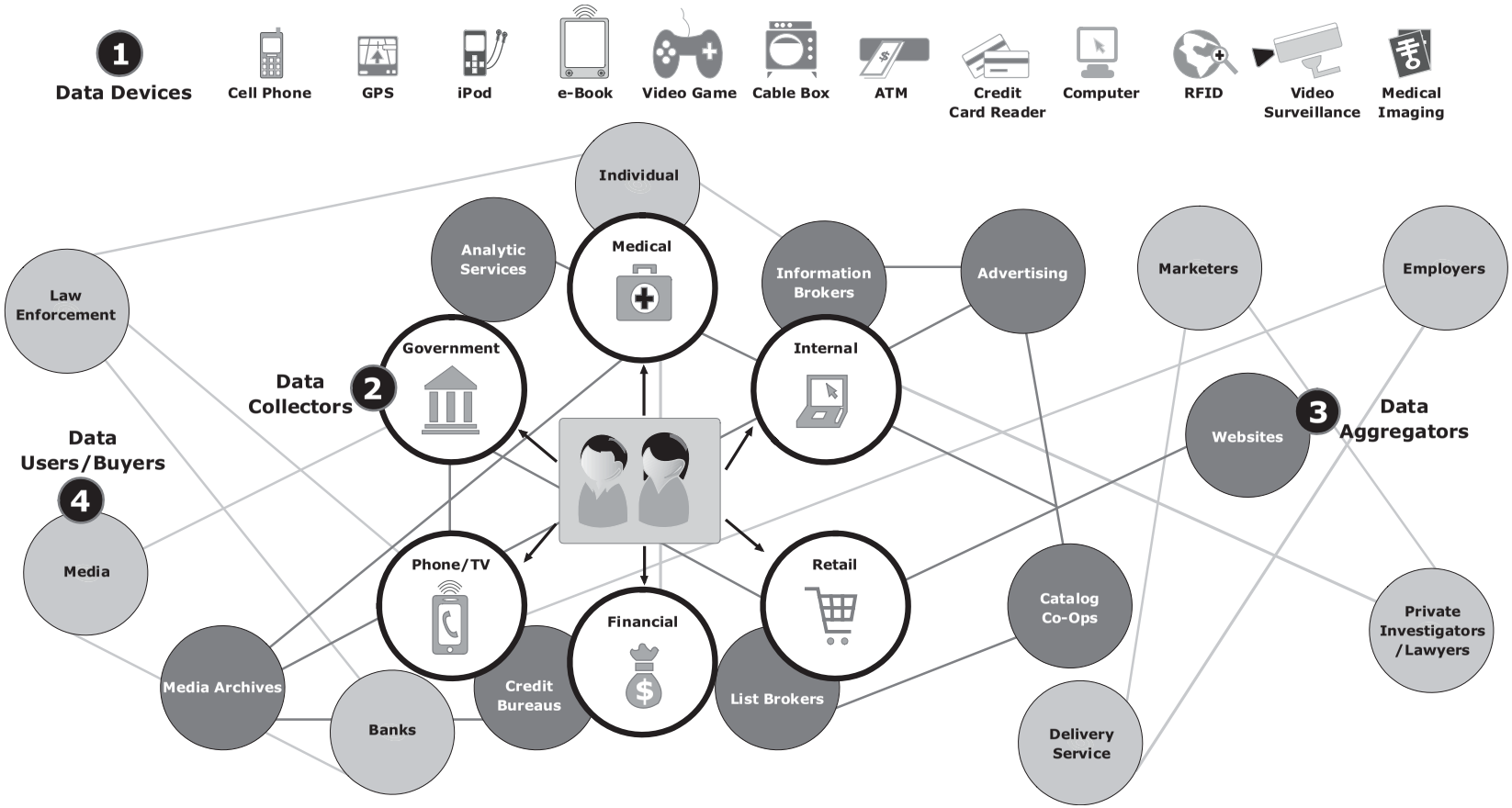


Figure 1-3: Big data ecosystem

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Traditional IT infrastructure and data processing tools and methodologies are inadequate to handle the volume, variety, dynamism, and complexity of big data. Analyzing big data in real time requires new techniques, architectures, and tools that provide high performance, massively parallel processing (MPP) data platforms, and advanced analytics on the data sets.

Data science is an emerging discipline, which enables organizations to derive business value from big data. Data science represents the synthesis of several existing disciplines, such as statistics, math, data visualization, and computer science to enable data scientists to develop advanced algorithms for the purpose of analyzing vast amounts of information to drive new value and make more data-driven decisions.

Several industries and markets currently looking to employ data science techniques include medical and scientific research, health care, public administration, fraud detection, social media, banks, insurance companies, and other digital information-based entities that benefit from the analytics of big data.

1.1.4 Information

Data, whether structured or unstructured, does not fulfill any purpose for individuals or businesses unless it is presented in a meaningful form. *Information* is the intelligence and knowledge derived from data.

Businesses analyze raw data to identify meaningful trends. On the basis of these trends, a company can plan or modify its strategy. For example, a retailer identifies customers' preferred products and brand names by analyzing their purchase patterns and maintaining an inventory of those products. Effective data analysis not only extends its benefits to existing businesses, but also creates the potential for new business opportunities by using the information in creative ways.

1.1.5 Storage

Data created by individuals or businesses must be stored so that it is easily accessible for further processing. In a computing environment, devices designed for storing data are termed *storage devices* or simply *storage*. The type of storage used varies based on the type of data and the rate at which it is created and used. Devices, such as a media card in a cell phone or digital camera, DVDs, CD-ROMs, and disk drives in personal computers are examples of storage devices.

Businesses have several options available for storing data, including internal hard disks, external disk arrays, and tapes.

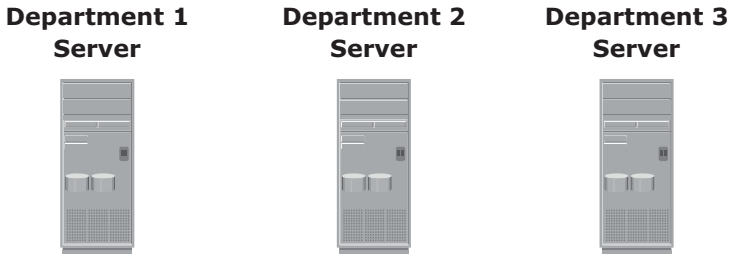
1.2 Evolution of Storage Architecture

Historically, organizations had centralized computers (mainframes) and information storage devices (tape reels and disk packs) in their data center. The evolution of open systems, their affordability, and ease of deployment made it

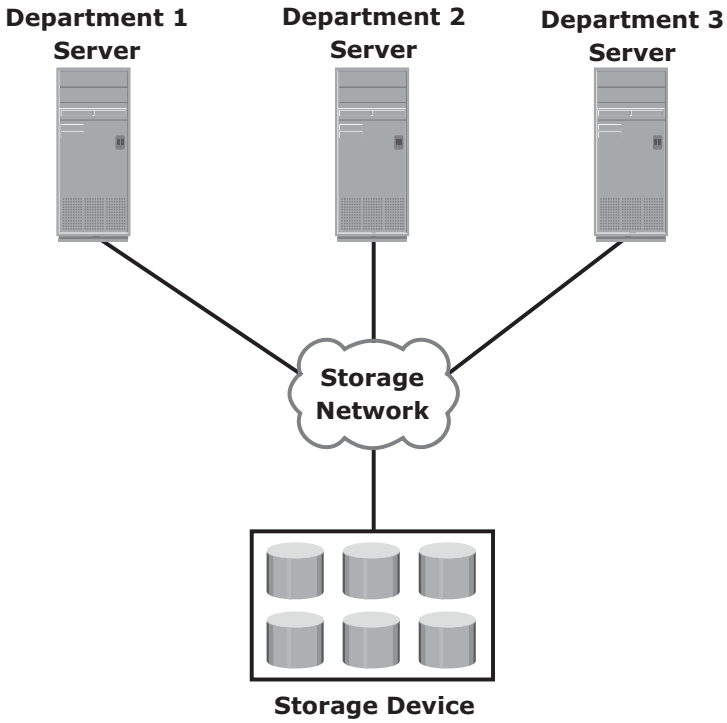
This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

possible for business units/departments to have their own servers and storage. In earlier implementations of open systems, the storage was typically internal to the server. These storage devices could not be shared with any other servers. This approach is referred to as *server-centric storage architecture* (see Figure 1-4 [a]). In this architecture, each server has a limited number of storage devices, and any administrative tasks, such as maintenance of the server or increasing storage capacity, might result in unavailability of information. The proliferation of departmental servers in an enterprise resulted in unprotected, unmanaged, fragmented islands of information and increased capital and operating expenses..



(a) Server-Centric Storage Architecture



(b) Information-Centric Storage Architecture

Figure 1-4: Evolution of storage architecture

To overcome these challenges, storage evolved from server-centric to *information-centric architecture* (see Figure 1-4 [b]). In this architecture, storage devices are managed centrally and independent of servers. These centrally-managed

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

storage devices are shared with multiple servers. When a new server is deployed in the environment, storage is assigned from the same shared storage devices to that server. The capacity of shared storage can be increased dynamically by adding more storage devices without impacting information availability. In this architecture, information management is easier and cost-effective.

Storage technology and architecture continue to evolve, which enables organizations to consolidate, protect, optimize, and leverage their data to achieve the highest return on information assets.

1.3 Data Center Infrastructure

Organizations maintain data centers to provide centralized data-processing capabilities across the enterprise. Data centers house and manage large amounts of data. The data center infrastructure includes hardware components, such as computers, storage systems, network devices, and power backups; and software components, such as applications, operating systems, and management software. It also includes environmental controls, such as air conditioning, fire suppression, and ventilation.

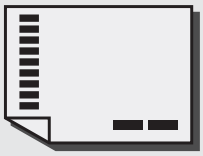
Large organizations often maintain more than one data center to distribute data processing workloads and provide backup if a disaster occurs.

1.3.1 Core Elements of a Data Center

Five core elements are essential for the functionality of a data center:

- **Application:** A computer program that provides the logic for computing operations
- **Database management system (DBMS):** Provides a structured way to store data in logically organized tables that are interrelated
- **Host or compute:** A computing platform (hardware, firmware, and software) that runs applications and databases
- **Network:** A data path that facilitates communication among various networked devices
- **Storage:** A device that stores data persistently for subsequent use

These core elements are typically viewed and managed as separate entities, but all the elements must work together to address data-processing requirements.



In this book, host, compute, and server are used interchangeably to represent the element that runs applications.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Figure 1-5 shows an example of an online order transaction system that involves the five core elements of a data center and illustrates their functionality in a business process.

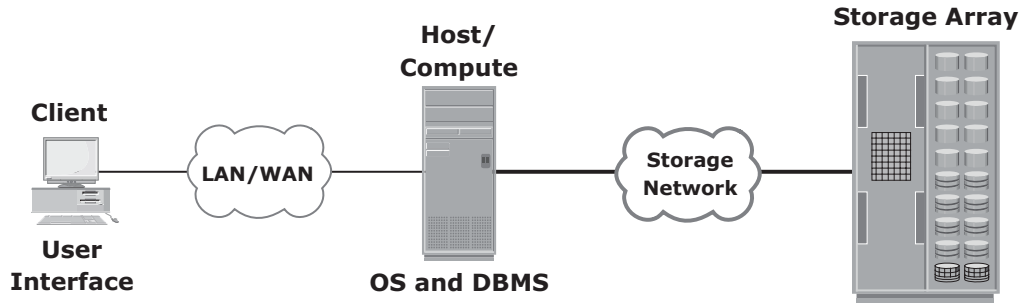


Figure 1-5: Example of an online order transaction system

A customer places an order through a client machine connected over a LAN/WAN to a host running an order-processing application. The client accesses the DBMS on the host through the application to provide order-related information, such as the customer name, address, payment method, products ordered, and quantity ordered.

The DBMS uses the host operating system to write this data to the physical disks in the storage array. The storage networks provide the communication link between the host and the storage array and transports the request to read or write data between them. The storage array, after receiving the read or write request from the host, performs the necessary operations to store the data on physical disks.

1.3.2 Key Characteristics of a Data Center

Uninterrupted operation of data centers is critical to the survival and success of a business. Organizations must have a reliable infrastructure that ensures that data is accessible at all times. Although the characteristics shown in Figure 1-6 are applicable to all elements of the data center infrastructure, the focus here is on storage systems. This book covers the various technologies and solutions to meet these requirements.

- **Availability:** A data center should ensure the availability of information when required. Unavailability of information could cost millions of dollars per hour to businesses, such as financial services, telecommunications, and e-commerce.
- **Security:** Data centers must establish policies, procedures, and core element integration to prevent unauthorized access to information.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

- **Scalability:** Business growth often requires deploying more servers, new applications, and additional databases. Data center resources should scale based on requirements, without interrupting business operations.
- **Performance:** All the elements of the data center should provide optimal performance based on the required service levels.
- **Data integrity:** Data integrity refers to mechanisms, such as error correction codes or parity bits, which ensure that data is stored and retrieved exactly as it was received.
- **Capacity:** Data center operations require adequate resources to store and process large amounts of data, efficiently. When capacity requirements increase, the data center must provide additional capacity without interrupting availability or with minimal disruption. Capacity may be managed by reallocating the existing resources or by adding new resources.
- **Manageability:** A data center should provide easy and integrated management of all its elements. Manageability can be achieved through automation and reduction of human (manual) intervention in common tasks.

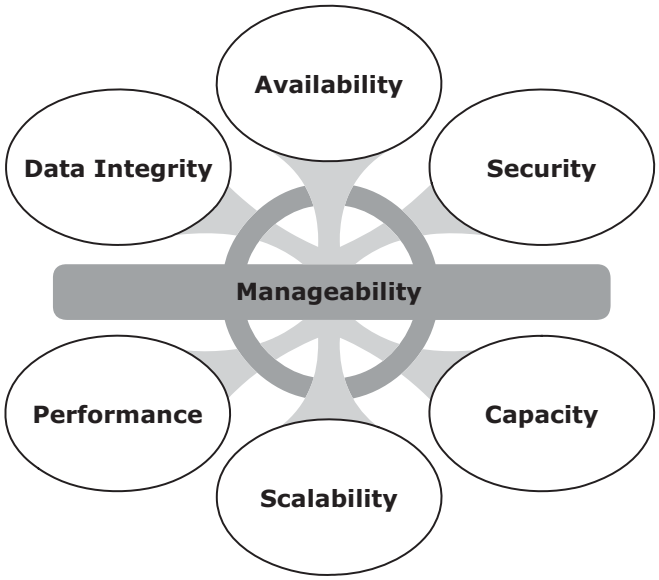


Figure 1-6: Key characteristics of a data center

1.3.3 Managing a Data Center

Managing a data center involves many tasks. The key management activities include the following:

- **Monitoring:** It is a continuous process of gathering information on various elements and services running in a data center. The aspects of a data

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

center that are monitored include security, performance, availability, and capacity.

- **Reporting:** It is done periodically on resource performance, capacity, and utilization. Reporting tasks help to establish business justifications and chargeback of costs associated with data center operations.
- **Provisioning:** It is a process of providing the hardware, software, and other resources required to run a data center. Provisioning activities primarily include resources management to meet capacity, availability, performance, and security requirements.

Virtualization and cloud computing have dramatically changed the way data center infrastructure resources are provisioned and managed. Organizations are rapidly deploying virtualization on various elements of data centers to optimize their utilization. Further, continuous cost pressure on IT and on-demand data processing requirements have resulted in the adoption of cloud computing.

1.4 Virtualization and Cloud Computing

Virtualization is a technique of abstracting physical resources, such as compute, storage, and network, and making them appear as logical resources. Virtualization has existed in the IT industry for several years and in different forms. Common examples of virtualization are virtual memory used on compute systems and partitioning of raw disks.

Virtualization enables pooling of physical resources and providing an aggregated view of the physical resource capabilities. For example, storage virtualization enables multiple pooled storage devices to appear as a single large storage entity. Similarly, by using compute virtualization, the CPU capacity of the pooled physical servers can be viewed as the aggregation of the power of all CPUs (in megahertz). Virtualization also enables centralized management of pooled resources.

Virtual resources can be created and provisioned from the pooled physical resources. For example, a virtual disk of a given capacity can be created from a storage pool or a virtual server with specific CPU power and memory can be configured from a compute pool. These virtual resources share pooled physical resources, which improves the utilization of physical IT resources. Based on business requirements, capacity can be added to or removed from the virtual resources without any disruption to applications or users. With improved utilization of IT assets, organizations save the costs associated with procurement and

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

management of new physical resources. Moreover, fewer physical resources means less space and energy, which leads to better economics and green computing.

In today's fast-paced and competitive environment, organizations must be agile and flexible to meet changing market requirements. This leads to rapid expansion and upgrade of resources while meeting shrinking or stagnant IT budgets. *Cloud computing*, addresses these challenges efficiently. Cloud computing enables individuals or businesses to use IT resources as a service over the network. It provides highly scalable and flexible computing that enables provisioning of resources on demand. Users can scale up or scale down the demand of computing resources, including storage capacity, with minimal management effort or service provider interaction. Cloud computing empowers self-service requesting through a fully automated request-fulfillment process. Cloud computing enables consumption-based metering; therefore, consumers pay only for the resources they use, such as CPU hours used, amount of data transferred, and gigabytes of data stored.

Cloud infrastructure is usually built upon virtualized data centers, which provide resource pooling and rapid provisioning of resources. Information storage in virtualized and cloud environments is detailed later in the book.

Summary

This chapter described the importance of data, information, and storage infrastructure. Meeting today's storage needs begins with understanding the type of data, its value, and key attributes of a data center.

The evolution of storage architecture and the core elements of a data center covered in this chapter provided the foundation for information storage and management. The emergence of virtualization has provided the opportunity to transform classic data centers into virtualized data centers. Cloud computing is further changing the way IT resources are provisioned and consumed.

The subsequent chapters in the book provide comprehensive details on various aspects of information storage and management in both classic and virtualized environments. It begins with describing the core elements of a data center with a focus on storage systems and RAID (covered in Chapters 2, 3, and 4). Chapters 5 through 8 of this book detail various storage networking technologies, such as storage area network (SAN), network attached storage (NAS), and object-based and unified storage. Chapters 9 through 12 cover various business continuity solutions, such as backup and replication, along with archival technologies. Chapter 13 introduces cloud infrastructure and services. Chapters 14 and 15 describe securing and managing storage in traditional and virtualized environments.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

EXERCISES

1. What is structured and unstructured data? Research the challenges of storing and managing unstructured data.
2. Discuss the benefits of information-centric storage architecture over server-centric storage architecture.
3. What are the attributes of big data? Research and prepare a presentation on big data analytics.
4. Research how businesses use their information assets to derive competitive advantage and new business opportunities.
5. Research and prepare a presentation on personal data management.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Chapter 2
Data Center Environment

KEY CONCEPTS

- Application, DBMS, Host, Connectivity, and Storage
- Application Virtualization
- File System and Volume Manager
- Compute, Desktop, and Memory Virtualization
- Storage Media
- Disk Drive Components
- Zoned Bit Recording
- Logical Block Addressing
- Flash Drives

Today, data centers are essential and integral parts of any business, whether small, medium, or large in size. The core elements of a data center are host, storage, connectivity (or network), applications, and DBMS that are managed centrally. These elements work together to process and store data. With the evolution of virtualization, data centers have also evolved from a classic data center to a virtualized data center (VDC). In a VDC, physical resources from a classic data center are pooled together and provided as virtual resources. This abstraction hides the complexity and limitation of physical resources from the user. By consolidating IT resources using virtualization, organizations can optimize their infrastructure utilization and reduce the total cost of owning an infrastructure. Moreover, in a VDC, virtual resources are created using software that enables faster deployment, compared to deploying physical resources in classic data centers. This chapter covers all the key components of a data center, including virtualization at compute, memory, desktop, and application. Storage and network virtualization is discussed later in the book.

With the increase in the criticality of information assets to businesses, storage — one of the core elements of a data center — is recognized as a distinct resource. Storage needs special focus and attention for its implementation and management. This chapter also focuses on storage subsystems and provides

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

details on components, geometry, and performance parameters of a disk drive. The connectivity between the host and storage facilitated by various technologies is also explained.

2.1 Application

An *application* is a computer program that provides the logic for computing operations. The application sends requests to the underlying operating system to perform read/write (R/W) operations on the storage devices. Applications can be layered on the database, which in turn uses the OS services to perform R/W operations on the storage devices. Applications deployed in a data center environment are commonly categorized as business applications, infrastructure management applications, data protection applications, and security applications. Some examples of these applications are e-mail, enterprise resource planning (ERP), decision support system (DSS), resource management, backup, authentication and antivirus applications, and so on.

The characteristics of I/Os (Input/Output) generated by the application influence the overall performance of storage system and storage solution designs. For more information on application I/O characteristics, refer to Appendix A.

APPLICATION VIRTUALIZATION

Application virtualization **breaks the dependency between the application and the underlying platform (OS and hardware). Application virtualization encapsulates the application and the required OS resources within a virtualized container. This technology provides the ability to deploy applications without making any change to the underlying OS, file system, or registry of the computing platform on which they are deployed. Because virtualized applications run in an isolated environment, the underlying OS and other applications are protected from potential corruptions. There are many scenarios in which conflicts might arise if multiple applications or multiple versions of the same application are installed on the same computing platform. Application virtualization eliminates this conflict by isolating different versions of an application and the associated O/S resources.**

2.2 Database Management System (DBMS)

A database is a structured way to store data in logically organized tables that are interrelated. A database helps to optimize the storage and retrieval of data. A DBMS controls the creation, maintenance, and use of a database. The DBMS

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

processes an application's request for data and instructs the operating system to transfer the appropriate data from the storage.

2.3 Host (Compute)

Users store and retrieve data through applications. The computers on which these applications run are referred to as *hosts* or *compute systems*. Hosts can be physical or virtual machines. A compute virtualization software enables creating virtual machines on top of a physical compute infrastructure. Compute virtualization and virtual machines are discussed later in this chapter. Examples of physical hosts include desktop computers, servers or a cluster of servers, laptops, and mobile devices. A host consists of CPU, memory, I/O devices, and a collection of software to perform computing operations. This software includes the operating system, file system, logical volume manager, device drivers, and so on. This software can be installed as separate entities or as part of the operating system.

The CPU consists of four components: Arithmetic Logic Unit (ALU), control unit, registers, and L1 cache. There are two types of memory on a host, Random Access Memory (RAM) and Read-Only Memory (ROM). I/O devices enable communication with a host. Examples of I/O devices are keyboard, mouse, monitor, etc.

Software runs on a host and enables processing of input and output (I/O) data. The following section details various software components that are essential parts of a host system.

2.3.1 Operating System

In a traditional computing environment, an *operating system* controls all aspects of computing. It works between the application and the physical components of a compute system. One of the services it provides to the application is data access. The operating system also monitors and responds to user actions and the environment. It organizes and controls hardware components and manages the allocation of hardware resources. It provides basic security for the access and usage of all managed resources. An operating system also performs basic storage management tasks while managing other underlying components, such as the file system, volume manager, and device drivers.

In a virtualized compute environment, the virtualization layer works between the operating system and the hardware resources. Here the OS might work differently based on the type of compute virtualization implemented. In a typical implementation, the OS works as a guest and performs only the activities related to application interaction. In this case, hardware management functions are handled by the virtualization layer.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Memory Virtualization

Memory has been, and continues to be, an expensive component of a host. It determines both the size and number of applications that can run on a host. *Memory virtualization* enables multiple applications and processes, whose aggregate memory requirement is greater than the available physical memory, to run on a host without impacting each other.

Memory virtualization is an operating system feature that virtualizes the physical memory (RAM) of a host. It creates virtual memory with an address space larger than the physical memory space present in the compute system. The virtual memory encompasses the address space of the physical memory and part of the disk storage. The operating system utility that manages the virtual memory is known as the *virtual memory manager* (VMM). The VMM manages the virtual-to-physical memory mapping and fetches data from the disk storage when a process references a virtual address that points to data at the disk storage. The space used by the VMM on the disk is known as a swap space. A *swap space* (also known as *page file* or *swap file*) is a portion of the disk drive that appears to be physical memory to the operating system.

In a virtual memory implementation, the memory of a system is divided into contiguous blocks of fixed-size pages. A process known as *paging* moves inactive physical memory pages onto the swap file and brings them back to the physical memory when required. This enables efficient use of the available physical memory among different applications. The operating system typically moves the least used pages into the swap file so that enough RAM is available for processes that are more active. Access to swap file pages is slower than access to physical memory pages because swap file pages are allocated on the disk drive, which is slower than physical memory.

2.3.2 Device Driver

A *device driver* is special software that permits the operating system to interact with a specific device, such as a printer, a mouse, or a disk drive. A device driver enables the operating system to recognize the device and to access and control devices. Device drivers are hardware-dependent and operating-system-specific.

2.3.3 Volume Manager

In the early days, disk drives appeared to the operating system as a number of continuous disk blocks. The entire disk drive would be allocated to the file system or other data entity used by the operating system or application. The

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

disadvantage was lack of flexibility. When a disk drive ran out of space, there was no easy way to extend the file system's size. Also, as the storage capacity of the disk drive increased, allocating the entire disk drive for the file system often resulted in underutilization of storage capacity.

The evolution of *Logical Volume Managers* (LVMs) enabled dynamic extension of file system capacity and efficient storage management. The LVM is software that runs on the compute system and manages logical and physical storage. LVM is an intermediate layer between the file system and the physical disk. It can partition a larger-capacity disk into virtual, smaller-capacity volumes (the process is called *partitioning*) or aggregate several smaller disks to form a larger virtual volume. (The process is called *concatenation*.) These volumes are then presented to applications.

Disk partitioning was introduced to improve the flexibility and utilization of disk drives. In partitioning, a disk drive is divided into logical containers called *logical volumes* (LVs) (see Figure 2-1). For example, a large physical drive can be partitioned into multiple LVs to maintain data according to the file system and application requirements. The partitions are created from groups of contiguous cylinders when the hard disk is initially set up on the host. The host's file system accesses the logical volumes without any knowledge of partitioning and physical structure of the disk.

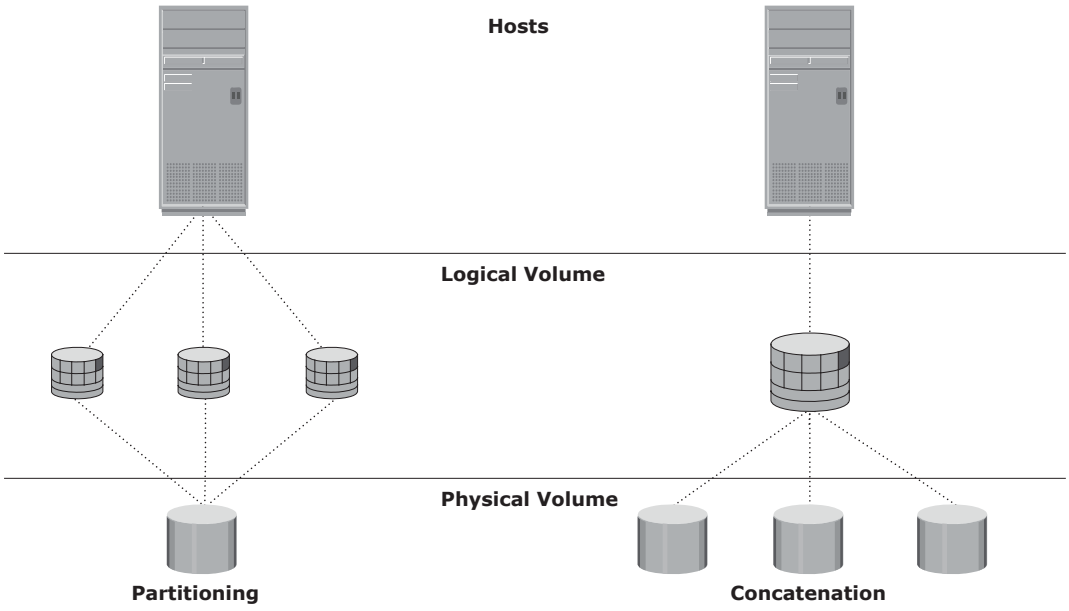


Figure 2-1: Disk partitioning and concatenation

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Concatenation is the process of grouping several physical drives and presenting them to the host as one big logical volume (see Figure 2-1).

The LVM provides optimized storage access and simplifies storage resource management. It hides details about the physical disk and the location of data on the disk. It enables administrators to change the storage allocation even when the application is running.

The basic LVM components are physical volumes, volume groups, and logical volumes. In LVM terminology, each physical disk connected to the host system is a *physical volume* (PV). The LVM converts the physical storage provided by the physical volumes to a logical view of storage, which is then used by the operating system and applications. A *volume group* is created by grouping together one or more physical volumes. A unique *physical volume identifier* (PVID) is assigned to each physical volume when it is initialized for use by the LVM. Physical volumes can be added or removed from a volume group dynamically. They cannot be shared between different volume groups, which means that the entire physical volume becomes part of a volume group. Each physical volume is partitioned into equal-sized data blocks called *physical extents* when the volume group is created.

Logical volumes are created within a given volume group. A logical volume can be thought of as a disk partition, whereas the volume group itself can be thought of as a disk. A volume group can have a number of logical volumes. The size of a logical volume is based on a multiple of the physical extents.

The logical volume appears as a physical device to the operating system. A logical volume is made up of noncontiguous physical extents and may span multiple physical volumes. A file system is created on a logical volume. These logical volumes are then assigned to the application. A logical volume can also be mirrored to provide enhanced data availability.

2.3.4 File System

A *file* is a collection of related records or data stored as a unit with a name. A *file system* is a hierarchical structure of files. A file system enables easy access to data files residing within a disk drive, a disk partition, or a logical volume. A file system consists of logical structures and software routines that control access to files. It provides users with the functionality to create, modify, delete, and access files. Access to files on the disks is controlled by the permissions assigned to the file by the owner, which are also maintained by the file system.

A file system organizes data in a structured hierarchical manner via the use of directories, which are containers for storing pointers to multiple files. All file systems maintain a pointer map to the directories, subdirectories, and files that are part of the file system. Examples of common file systems are:

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

- FAT 32 (File Allocation Table) for Microsoft Windows
- NT File System (NTFS) for Microsoft Windows
- UNIX File System (UFS) for UNIX
- Extended File System (EXT2/3) for Linux

Apart from the files and directories, the file system also includes a number of other related records, which are collectively called the *metadata*. For example, the metadata in a UNIX environment consists of the *superblock*, the *inodes*, and the list of data blocks free and in use. The metadata of a file system must be consistent for the file system to be considered healthy.

A superblock contains important information about the file system, such as the file system type, creation and modification dates, size, and layout. It also contains the count of available resources (such as the number of free blocks, inodes, and so on) and a flag indicating the mount status of the file system. An inode is associated with every file and directory and contains information such as the file length, ownership, access privileges, time of last access/modification, number of links, and the address of the data.

A file system *block* is the smallest "unit" allocated for storing data. Each file system block is a contiguous area on the physical disk. The block size of a file system is fixed at the time of its creation. The file system size depends on the block size and the total number of file system blocks. A file can span multiple file system blocks because most files are larger than the predefined block size of the file system. File system blocks cease to be contiguous and become fragmented when new blocks are added or deleted. Over time, as files grow larger, the file system becomes increasingly fragmented.

The following list shows the process of mapping user files to the disk storage subsystem with an LVM (see Figure 2-2):

1. Files are created and managed by users and applications.
2. These files reside in the file systems.
3. The file systems are mapped to file system blocks.
4. The file system blocks are mapped to logical extents of a logical volume.
5. These logical extents in turn are mapped to the disk physical extents either by the operating system or by the LVM.
6. These physical extents are mapped to the disk sectors in a storage subsystem.

If there is no LVM, then there are no logical extents. Without LVM, file system blocks are directly mapped to disk sectors.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

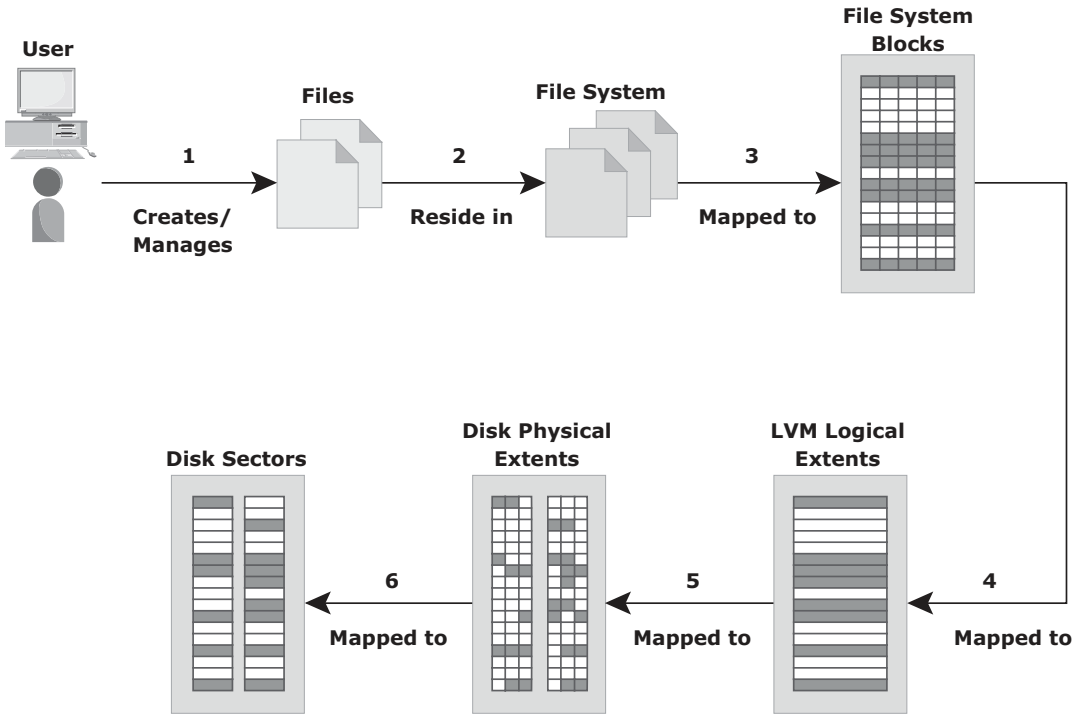
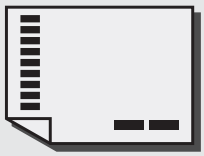


Figure 2-2: Process of mapping user files to disk storage

The *file system tree* starts with the *root directory*. The root directory has a number of subdirectories. A file system should be mounted before it can be used.



The system utility `fsck` is run to check file system consistency in UNIX and Linux hosts. An example of the file system in an inconsistent state is when the file system has outstanding changes and the computer system crashes before the changes are committed to disk. At the time of booting, the `fsck` command first checks for consistency of file systems for a successful boot. If the file systems are found to be consistent, the command checks the consistency of all other file systems. If any file system is found to be inconsistent, it is not mounted. The inconsistent file system might be repaired automatically by the `fsck` command or might require user interaction for confirmation of corrective actions. `CHKDSK` is the command used on DOS, OS/2, and Microsoft Windows operating systems.

A file system can be either a journaling file system or a nonjournaling file system. *Nonjournaling file systems* cause a potential loss of files because they use separate writes to update their data and metadata. If the system crashes during the write process, the metadata or data might be lost or corrupted. When the

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

system reboots, the file system attempts to update the metadata structures by examining and repairing them. This operation takes a long time on large file systems. If there is insufficient information to re-create the wanted or original structure, the files might be misplaced or lost, resulting in corrupted file systems.

A *journaling file system* uses a separate area called a *log* or *journal*. This journal might contain all the data to be written (*physical journal*) or just the metadata to be updated (*logical journal*). Before changes are made to the file system, they are written to this separate area. After the journal has been updated, the operation on the file system can be performed. If the system crashes during the operation, there is enough information in the log to "replay" the log record and complete the operation. Journaling results in a quick file system check because it looks only at the active, most recently accessed parts of a large file system. In addition, because information about the pending operation is saved, the risk of files being lost is reduced.

A disadvantage of journaling file systems is that they are slower than other file systems. This slowdown is the result of the extra operations that have to be performed on the journal each time the file system is changed. However, the much shortened time for file system checks and the file system integrity provided by journaling far outweighs its disadvantage. Nearly all file system implementations today use journaling.

Dedicated file servers may be installed to manage and share a large number of files over a network. These file servers support multiple file systems and use file-sharing protocols specific to the operating system — for example, NFS and CIFS. These protocols are detailed in Chapter 7.

2.3.5 Compute Virtualization

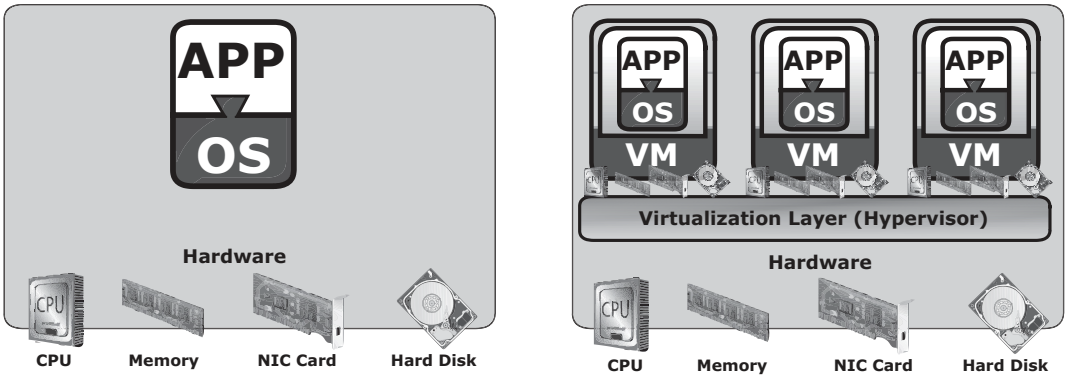
Compute virtualization is a technique for masking or abstracting the physical hardware from the operating system. It enables multiple operating systems to run concurrently on single or clustered physical machines. This technique enables creating portable virtual compute systems called *virtual machines* (VMs). Each VM runs an operating system and application instance in an isolated manner. Compute virtualization is achieved by a virtualization layer that resides between the hardware and virtual machines. This layer is also called the *hypervisor*. The hypervisor provides hardware resources, such as CPU, memory, and network to all the virtual machines. Within a physical server, a large number of virtual machines can be created depending on the hardware capabilities of the physical server.

A virtual machine is a logical entity but appears like a physical host to the operating system, with its own CPU, memory, network controller, and disks. However, all VMs share the same underlying physical hardware in an isolated manner. From a hypervisor perspective, virtual machines are discrete sets of files that include VM configuration file, data files, and so on.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Typically, a physical server often faces resource-conflict issues when two or more applications running on the server have conflicting requirements. For example, applications might need different values in the same registry entry, different versions of the same DLL, and so on. These issues are further compounded with an application's high-availability requirements. As a result, the servers are limited to serve only one application at a time, as shown in Figure 2-3 (a). This causes organizations to purchase new physical machines for every application they deploy, resulting in expensive and inflexible infrastructure. On the other hand, many applications do not take full advantage of the hardware capabilities available to them. Consequently, resources such as processors, memory, and storage remain underutilized. Compute virtualization enables users to overcome these challenges (see Figure 2-3 [b]) by allowing multiple operating systems and applications to run on a single physical machine. This technique significantly improves server utilization and provides server consolidation.



(a) Before Compute Virtualization

(b) After Compute Virtualization

Figure 2-3: Server virtualization

Server consolidation enables organizations to run their data center with fewer servers. This, in turn, cuts down the cost of new server acquisition, reduces operational cost, and saves data center floor and rack space. Creation of VMs takes less time compared to a physical server setup; organizations can provision servers faster and with ease. Individual VMs can be restarted, upgraded, or even crashed, without affecting the other VMs on the same physical machine. Moreover, VMs can be copied or moved from one physical machine to another without causing application downtime. Nondisruptive migration of VMs is required for load balancing among physical machines, hardware maintenance, and availability purposes.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

DESKTOP VIRTUALIZATION

With the traditional desktop, the OS, applications, and user profiles are all tied to a specific piece of hardware. With legacy desktops, business productivity is impacted greatly when a client device is broken or lost. *Desktop virtualization* breaks the dependency between the hardware and its OS, applications, user profiles, and settings. This enables the IT staff to change, update, and deploy these elements independently. Desktops hosted at the data center run on virtual machines; users remotely access these desktops from a variety of client devices, such as laptops, desktops, and mobile devices (also called Thin devices). Application execution and data storage are performed centrally at the data center instead of at the client devices. Because desktops run as virtual machines within an organization's data center, it mitigates the risk of data leakage and theft. It also helps to perform centralized backup and simplifies compliance procedures. Virtual desktops are easy to maintain because it is simple to apply patches, deploy new applications and OS, and provision or remove users centrally.

2.4 Connectivity

Connectivity refers to the interconnection between hosts or between a host and peripheral devices, such as printers or storage devices. The discussion here focuses only on the connectivity between the host and the storage device. Connectivity and communication between host and storage are enabled using physical components and interface protocols.

2.4.1 Physical Components of Connectivity

The *physical components* of connectivity are the hardware elements that connect the host to storage. Three physical components of connectivity between the host and storage are the host interface device, port, and cable (Figure 2-4).

A *host interface device* or *host adapter* connects a host to other hosts and storage devices. Examples of host interface devices are host bus adapter (HBA) and network interface card (NIC). *Host bus adaptor* is an *application-specific integrated circuit* (ASIC) board that performs I/O interface functions between the host and storage, relieving the CPU from additional I/O processing workload. A host typically contains multiple HBAs.

A *port* is a specialized outlet that enables connectivity between the host and external devices. An HBA may contain one or more ports to connect the host

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

to the storage device. *Cables* connect hosts to internal or external devices using copper or fiber optic media.

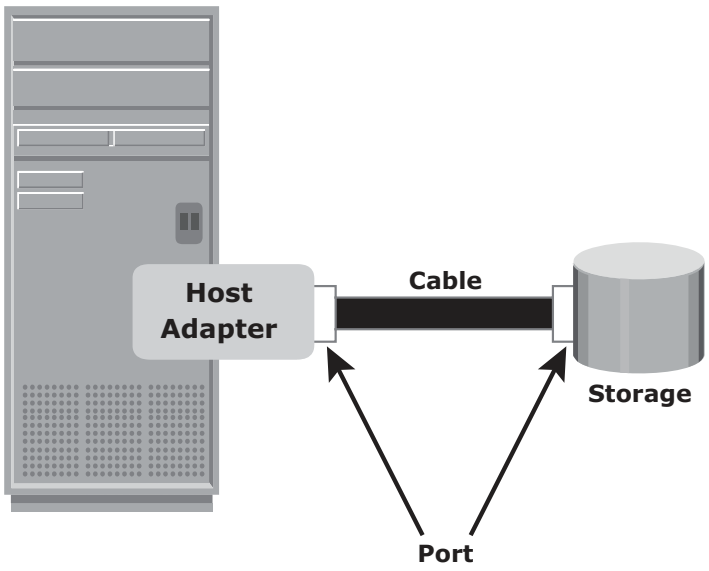


Figure 2-4: Physical components of connectivity

2.4.2 Interface Protocols

A *protocol* enables communication between the host and storage. Protocols are implemented using interface devices (or controllers) at both source and destination. The popular interface protocols used for host to storage communications are *Integrated Device Electronics/Advanced Technology Attachment* (IDE/ATA), *Small Computer System Interface* (SCSI), *Fibre Channel* (FC) and *Internet Protocol* (IP).

IDE/ATA and Serial ATA

IDE/ATA is a popular interface protocol standard used for connecting storage devices, such as disk drives and CD-ROM drives. This protocol supports parallel transmission and therefore is also known as Parallel ATA (PATA) or simply ATA. IDE/ATA has a variety of standards and names. The Ultra DMA/133 version of ATA supports a throughput of 133 MB per second. In a master-slave configuration, an ATA interface supports two storage devices per connector. However, if the performance of the drive is important, sharing a port between two devices is not recommended.

The serial version of this protocol supports single bit serial transmission and is known as Serial ATA (SATA). High performance and low cost SATA has largely replaced PATA in newer systems. SATA revision 3.0 provides a data transfer rate up to 6 Gb/s.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

SCSI and Serial SCSI

SCSI has emerged as a preferred connectivity protocol in high-end computers. This protocol supports parallel transmission and offers improved performance, scalability, and compatibility compared to ATA. However, the high cost associated with SCSI limits its popularity among home or personal desktop users. Over the years, SCSI has been enhanced and now includes a wide variety of related technologies and standards. SCSI supports up to 16 devices on a single bus and provides data transfer rates up to 640 MB/s (for the Ultra-640 version).

Serial attached SCSI (SAS) is a point-to-point serial protocol that provides an alternative to parallel SCSI. A newer version of serial SCSI (SAS 2.0) supports a data transfer rate up to 6 Gb/s. This book's Appendix B provides more details on the SCSI architecture and interface.

Fibre Channel

Fibre Channel is a widely used protocol for high-speed communication to the storage device. The Fibre Channel interface provides gigabit network speed. It provides a serial data transmission that operates over copper wire and optical fiber. The latest version of the FC interface (16FC) allows transmission of data up to 16 Gb/s. The FC protocol and its features are covered in more detail in Chapter 5.

Internet Protocol (IP)

IP is a network protocol that has been traditionally used for host-to-host traffic. With the emergence of new technologies, an IP network has become a viable option for host-to-storage communication. IP offers several advantages in terms of cost and maturity and enables organizations to leverage their existing IP-based network. iSCSI and FCIP protocols are common examples that leverage IP for host-to-storage communication. These protocols are detailed in Chapter 6.

2.5 Storage

Storage is a core component in a data center. A storage device uses magnetic, optic, or solid state media. Disks, tapes, and diskettes use magnetic media, whereas CD/DVD uses optical media for storage. Removable Flash memory or Flash drives are examples of solid state media.

In the past, *tapes* were the most popular storage option for backups because of their low cost. However, tapes have various limitations in terms of performance and management, as listed here:

- Data is stored on the tape linearly along the length of the tape. Search and retrieval of data are done sequentially, and it invariably takes several

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

seconds to access the data. As a result, random data access is slow and time-consuming. This limits tapes as a viable option for applications that require real-time, rapid access to data.

- In a shared computing environment, data stored on tape cannot be accessed by multiple applications simultaneously, restricting its use to one application at a time.
- On a tape drive, the read/write head touches the tape surface, so the tape degrades or wears out after repeated use.
- The storage and retrieval requirements of data from the tape and the overhead associated with managing the tape media are significant.

Due to these limitations and availability of low-cost disk drives, tapes are no longer a preferred choice as a backup destination for enterprise-class data centers.

Optical disc storage is popular in small, single-user computing environments. It is frequently used by individuals to store photos or as a backup medium on personal or laptop computers. It is also used as a distribution medium for small applications, such as games, or as a means to transfer small amounts of data from one computer system to another. Optical discs have limited capacity and speed, which limit the use of optical media as a business data storage solution.

The capability to *write once and read many* (WORM) is one advantage of optical disc storage. A CD-ROM is an example of a WORM device. Optical discs, to some degree, guarantee that the content has not been altered. Therefore, it can be used as a low-cost alternative for long-term storage of relatively small amounts of fixed content that do not change after it is created. Collections of optical discs in an array, called a *jukebox*, are still used as a fixed-content storage solution. Other forms of optical discs include CD-RW, Blu-ray disc, and other variations of DVD.

Disk drives are the most popular storage medium used in modern computers for storing and accessing data for performance-intensive, online applications. Disks support rapid access to random data locations. This means that data can be written or retrieved quickly for a large number of simultaneous users or applications. In addition, disks have a large capacity. Disk storage arrays are configured with multiple disks to provide increased capacity and enhanced performance.

Disk drives are accessed through predefined protocols, such as ATA, Serial ATA (SATA), SAS (Serial Attached SCSI), and FC. These protocols are implemented on the disk interface controllers. Earlier, disk interface controllers were implemented as separate cards, which were connected to the motherboard to provide communication with storage devices. Modern disk interface controllers are integrated with the disk drives; therefore, disk drives are known by the protocol interface they support, for example SATA disk, FC disk, and so on.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.6 Disk Drive Components

The key components of a hard disk drive are platter, spindle, read-write head, actuator arm assembly, and controller board (see Figure 2-5).

I/O operations in a HDD are performed by rapidly moving the arm across the rotating flat platters coated with magnetic particles. Data is transferred between the disk controller and magnetic platters through the read-write (R/W) head which is attached to the arm. Data can be recorded and erased on magnetic platters any number of times. Following sections detail the different components of the disk drive, the mechanism for organizing and storing data on disks, and the factors that affect disk performance.

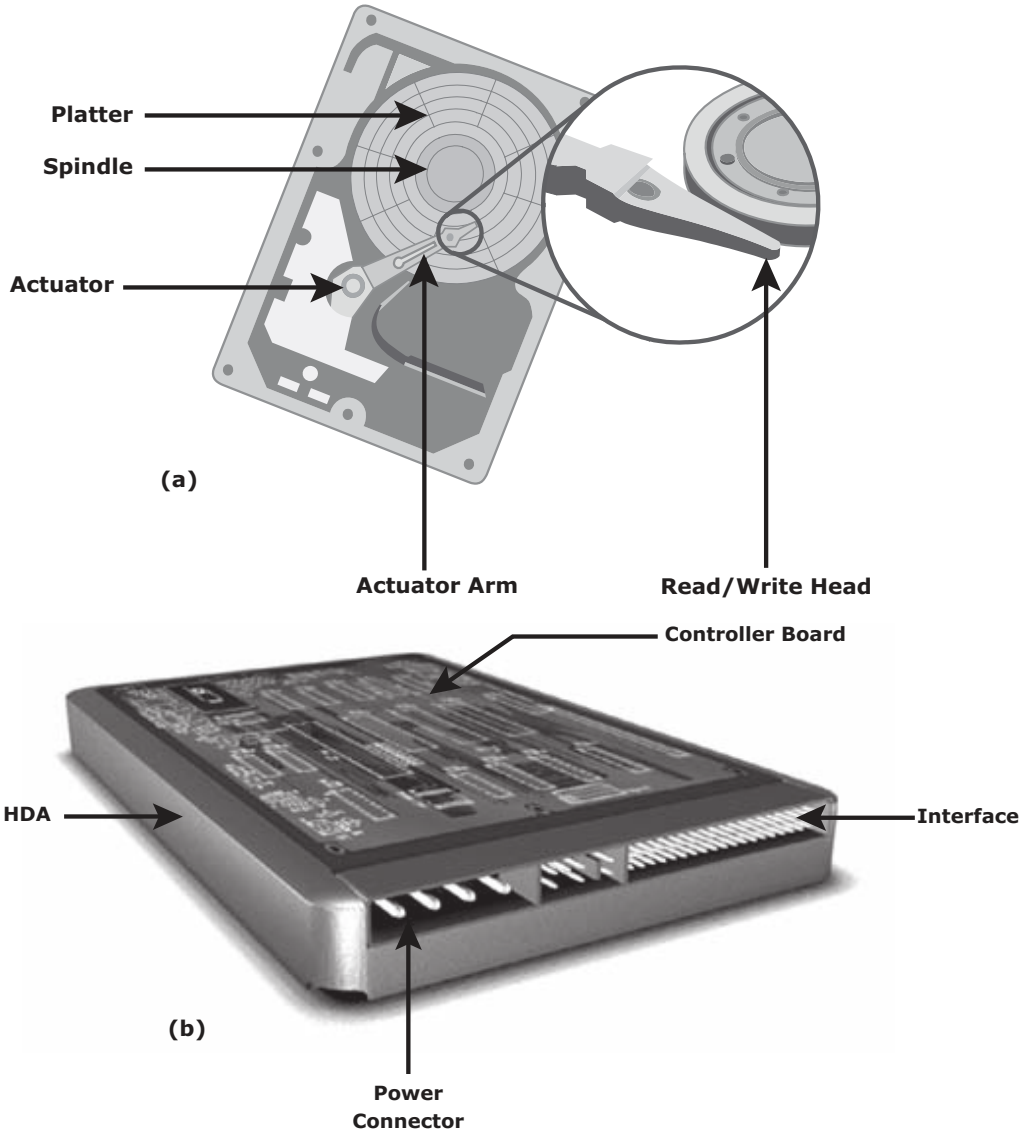


Figure 2-5: Disk drive components

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.6.1 Platter

A typical HDD consists of one or more flat circular disks called *platters* (Figure 2-6). The data is recorded on these platters in binary codes (`0`s and `1`s). The set of rotating platters is sealed in a case, called the *Head Disk Assembly* (HDA). A platter is a rigid, round disk coated with magnetic material on both surfaces (top and bottom). The data is encoded by polarizing the magnetic area, or domains, of the disk surface. Data can be written to or read from both surfaces of the platter. The number of platters and the storage capacity of each platter determine the total capacity of the drive.

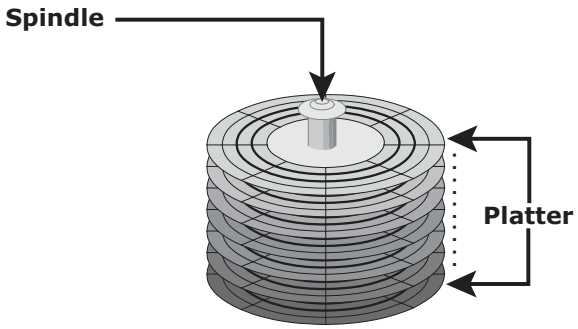


Figure 2-6: Spindle and platter

2.6.2 Spindle

A spindle connects all the platters (refer to Figure 2-6) and is connected to a motor. The motor of the spindle rotates with a constant speed.

The disk platter spins at a speed of several thousands of revolutions per minute (rpm). Common spindle speeds are 5,400 rpm, 7,200 rpm, 10,000 rpm, and 15,000 rpm. The speed of the platter is increasing with improvements in technology, although the extent to which it can be improved is limited.

2.6.3 Read/Write Head

Read/Write (R/W) heads, as shown in Figure 2-7, read and write data from or to platters. Drives have two R/W heads per platter, one for each surface of the platter. The R/W head changes the magnetic polarization on the surface of the platter when writing data. While reading data, the head detects the magnetic polarization on the surface of the platter. During reads and writes, the R/W head senses the magnetic polarization and never touches the surface of the platter. When the spindle is rotating, there is a microscopic air gap maintained between the R/W heads and the platters, known as the *head flying height*. This air gap is removed when the spindle stops rotating and the R/W head rests on a special area on the platter near the spindle. This area is called the *landing*

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

zone. The landing zone is coated with a lubricant to reduce friction between the head and the platter.

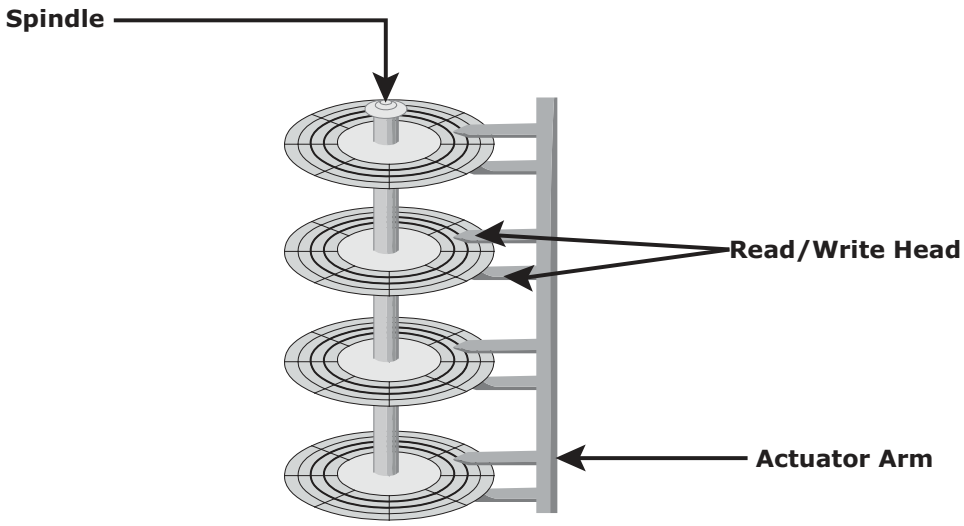


Figure 2-7: Actuator arm assembly

The logic on the disk drive ensures that heads are moved to the landing zone before they touch the surface. If the drive malfunctions and the R/W head accidentally touches the surface of the platter outside the landing zone, a *head crash* occurs. In a head crash, the magnetic coating on the platter is scratched and may cause damage to the R/W head. A head crash generally results in data loss.

2.6.4 Actuator Arm Assembly

R/W heads are mounted on the *actuator arm assembly*, which positions the R/W head at the location on the platter where the data needs to be written or read (refer to Figure 2-7). The R/W heads for all platters on a drive are attached to one actuator arm assembly and move across the platters simultaneously.

2.6.5 Drive Controller Board

The controller (refer to Figure 2-5 [b]) is a printed circuit board, mounted at the bottom of a disk drive. It consists of a microprocessor, internal memory, circuitry, and firmware. The firmware controls the power to the spindle motor and the speed of the motor. It also manages the communication between the drive and the host. In addition, it controls the R/W operations by moving the actuator arm and switching between different R/W heads, and performs the optimization of data access.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.6.6 Physical Disk Structure

Data on the disk is recorded on *tracks*, which are concentric rings on the platter around the spindle, as shown in Figure 2-8. The tracks are numbered, starting from zero, from the outer edge of the platter. The number of *tracks per inch* (TPI) on the platter (or the *track density*) measures how tightly the tracks are packed on a platter.

Each track is divided into smaller units called *sectors*. A sector is the smallest, individually addressable unit of storage. The track and sector structure is written on the platter by the drive manufacturer using a low-level formatting operation. The number of sectors per track varies according to the drive type. The first personal computer disks had 17 sectors per track. Recent disks have a much larger number of sectors on a single track. There can be thousands of tracks on a platter, depending on the physical dimensions and recording density of the platter.

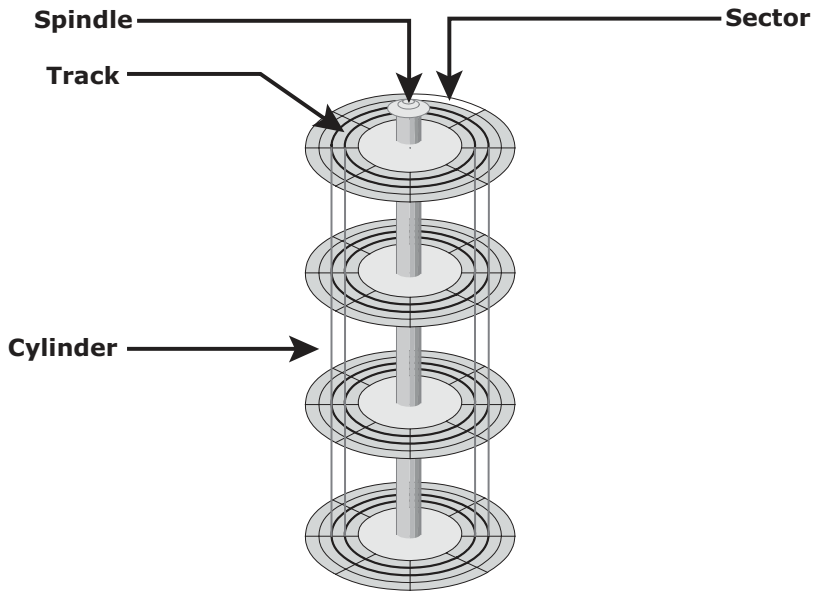


Figure 2-8: Disk structure: sectors, tracks, and cylinders

Typically, a sector holds 512 bytes of user data, although some disks can be formatted with larger sector sizes. In addition to user data, a sector also stores other information, such as the sector number, head number or platter number, and track number. This information helps the controller to locate the data on the drive.

A cylinder is a set of identical tracks on both surfaces of each drive platter. The location of R/W heads is referred to by the cylinder number, not by the track number.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

DISK ADVERTISED CAPACITY VERSUS AVAILABLE CAPACITY

A difference exists between the advertised capacity of a disk and the actual space available for data storage. For example, a disk advertised as 500 GB has only 465.7 GB of user-data capacity. The reason for this difference is that drive manufacturers use a base of 10 for the disk capacity, which means 1 kilobyte is equal to 1,000 bytes instead of 1,024 bytes; therefore, the actual available capacity of a disk is always less than the advertised capacity.

2.6.7 Zoned Bit Recording

Platters are made of concentric tracks; the outer tracks can hold more data than the inner tracks because the outer tracks are physically longer than the inner tracks. On older disk drives, the outer tracks had the same number of sectors as the inner tracks, so data density was low on the outer tracks. This was an inefficient use of the available space, as shown in Figure 2-9 (a).

Zoned bit recording uses the disk efficiently. As shown in Figure 2-9 (b), this mechanism groups tracks into zones based on their distance from the center of the disk. The zones are numbered, with the outermost zone being zone 0. An appropriate number of sectors per track are assigned to each zone, so a zone near the center of the platter has fewer sectors per track than a zone on the outer edge. However, tracks within a particular zone have the same number of sectors.

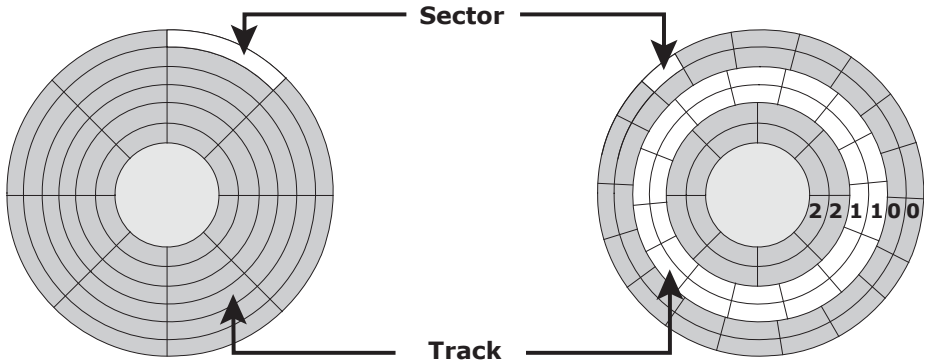
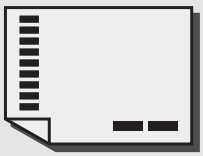


Figure 2-9: Zoned bit recording



The data transfer rate drops while accessing data from zones closer to the center of the platter. Applications that demand high performance should have their data on the outer zones of the platter.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.6.8 Logical Block Addressing

Earlier drives used physical addresses consisting of the *cylinder, head,* and *sector* (CHS) number to refer to specific locations on the disk, as shown in Figure 2-10 (a), and the host operating system had to be aware of the geometry of each disk used. *Logical block addressing* (LBA), as shown in Figure 2-10 (b), simplifies addressing by using a linear address to access physical blocks of data. The disk controller translates LBA to a CHS address, and the host needs to know only the size of the disk drive in terms of the number of blocks. The logical blocks are mapped to physical sectors on a 1:1 basis.

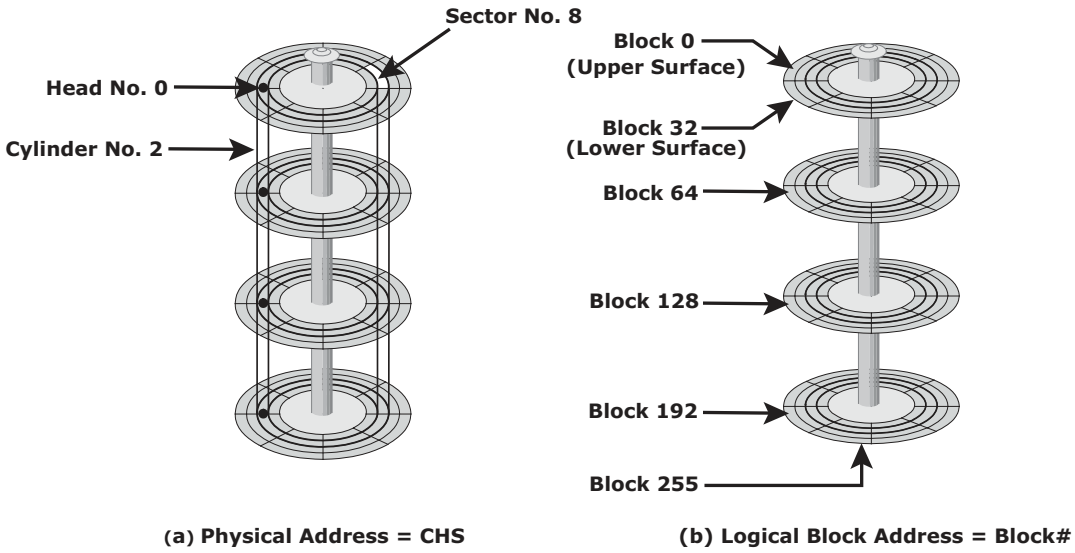


Figure 2-10: Physical address and logical block address

In Figure 2-10 (b), the drive shows eight sectors per track, eight heads, and four cylinders. This means a total of 8 × 8 × 4 = 256 blocks, so the block number ranges from 0 to 255. Each block has its own unique address. Assuming that the sector holds 512 bytes, a 500 GB drive with a formatted capacity of 465.7 GB has in excess of 976,000,000 blocks.

2.7 Disk Drive Performance

A disk drive is an electromechanical device that governs the overall performance of the storage system environment. The various factors that affect the performance of disk drives are discussed in this section.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.7.1 Disk Service Time

Disk service time is the time taken by a disk to complete an I/O request. Components that contribute to the service time on a disk drive are *seek time*, *rotational latency*, and *data transfer rate*.

Seek Time

The *seek time* (also called *access time*) describes the time taken to position the R/W heads across the platter with a radial movement (moving along the radius of the platter). In other words, it is the time taken to position and settle the arm and the head over the correct track. Therefore, the lower the seek time, the faster the I/O operation. Disk vendors publish the following seek time specifications:

- **Full Stroke:** The time taken by the R/W head to move across the entire width of the disk, from the innermost track to the outermost track.
- **Average:** The average time taken by the R/W head to move from one random track to another, normally listed as the time for one-third of a full stroke.
- **Track-to-Track:** The time taken by the R/W head to move between adjacent tracks.

Each of these specifications is measured in milliseconds. The seek time of a disk is typically specified by the drive manufacturer. The average seek time on a modern disk is typically in the range of 3 to 15 milliseconds. Seek time has more impact on the read operation of random tracks rather than adjacent tracks. To minimize the seek time, data can be written to only a subset of the available cylinders. This results in lower usable capacity than the actual capacity of the drive. For example, a 500 GB disk drive is set up to use only the first 40 percent of the cylinders and is effectively treated as a 200 GB drive. This is known as *short-stroking* the drive.

Rotational Latency

To access data, the actuator arm moves the R/W head over the platter to a particular track while the platter spins to position the requested sector under the R/W head. The time taken by the platter to rotate and position the data under the R/W head is called *rotational latency*. This latency depends on the rotation speed of the spindle and is measured in milliseconds. The average rotational latency is one-half of the time taken for a full rotation. Similar to the seek time,

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

rotational latency has more impact on the reading/writing of random sectors on the disk than on the same operations on adjacent sectors.

Average rotational latency is approximately 5.5 ms for a 5,400-rpm drive, and around 2.0 ms for a 15,000-rpm (or 250-rps revolution per second) drive as shown here:

> Average rotational latency for a 15,000 rpm (or 250 rps) drive = 0.5/250 = 2 milliseconds.

Data Transfer Rate

The *data transfer rate* (also called *transfer rate*) refers to the average amount of data per unit time that the drive can deliver to the HBA. It is important to first understand the process of read/write operations to calculate data transfer rates. In a *read operation*, the data first moves from disk platters to R/W heads; then it moves to the drive's internal *buffer*. Finally, data moves from the buffer through the interface to the host HBA. In a *write operation*, the data moves from the HBA to the internal buffer of the disk drive through the drive's interface. The data then moves from the buffer to the R/W heads. Finally, it moves from the R/W heads to the platters.

The data transfer rates during the R/W operations are measured in terms of internal and external transfer rates, as shown in Figure 2-11.

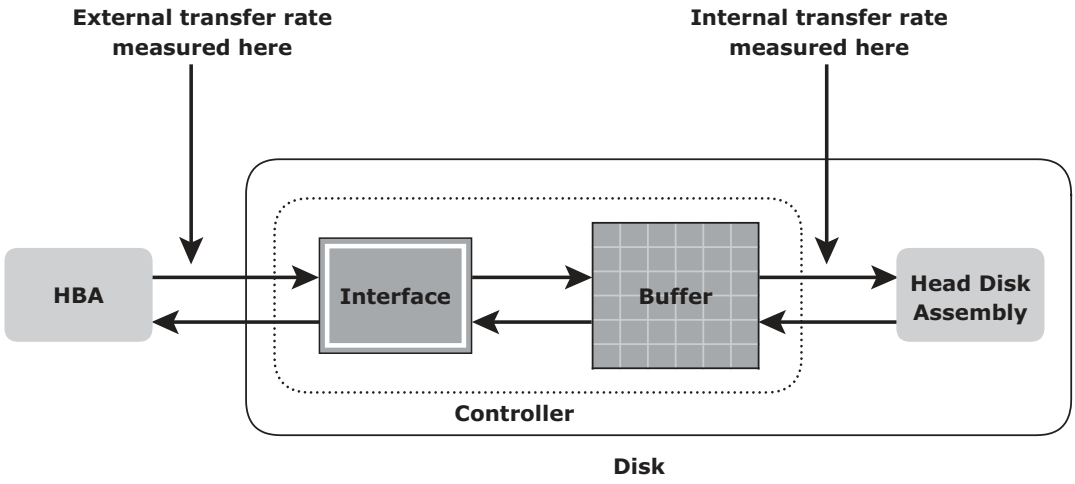


Figure 2-11: Data transfer rate

Internal transfer rate is the speed at which data moves from a platter's surface to the internal buffer (cache) of the disk. The internal transfer rate takes into account factors such as the seek time and rotational latency. *External transfer rate* is the rate at which data can move through the interface to the HBA. The external transfer rate is generally the advertised speed of the interface, such as 133 MB/s for ATA. The sustained external transfer rate is lower than the interface speed.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

2.7.2 Disk I/O Controller Utilization

Utilization of a disk I/O controller has a significant impact on the I/O response time. To understand this impact, consider that a disk can be viewed as a black box consisting of two elements:

- **Queue:** The location where an I/O request waits before it is processed by the I/O controller
- **Disk I/O Controller:** Processes I/Os waiting in the queue one by one

The I/O requests arrive at the controller at the rate generated by the application. This rate is also called the *arrival rate*. These requests are held in the I/O queue, and the I/O controller processes them one by one, as shown in Figure 2-12. The I/O arrival rate, the queue length, and the time taken by the I/O controller to process each request determines the I/O response time. If the controller is busy or heavily utilized, the queue size will be large and the response time will be high.

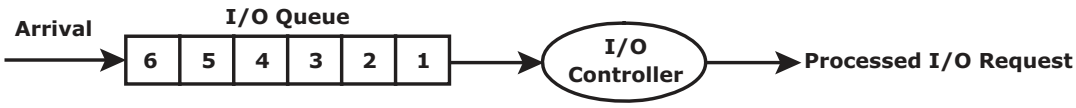


Figure 2-12: I/O processing

Based on the fundamental laws of disk drive performance, the relationship between controller utilization and average response time is given as

$$\text{Average response time } (T_R) = \text{Service time } (T_S) \,/\, (1 - \text{Utilization})$$

where T_S is the time taken by the controller to serve an I/O.

As the utilization reaches 100 percent — that is, as the I/O controller saturates — the response time is closer to infinity. In essence, the saturated component, or the bottleneck, forces the serialization of I/O requests, meaning that each I/O request must wait for the completion of the I/O requests that preceded it. Figure 2-13 shows a graph plotted between utilization and response time.

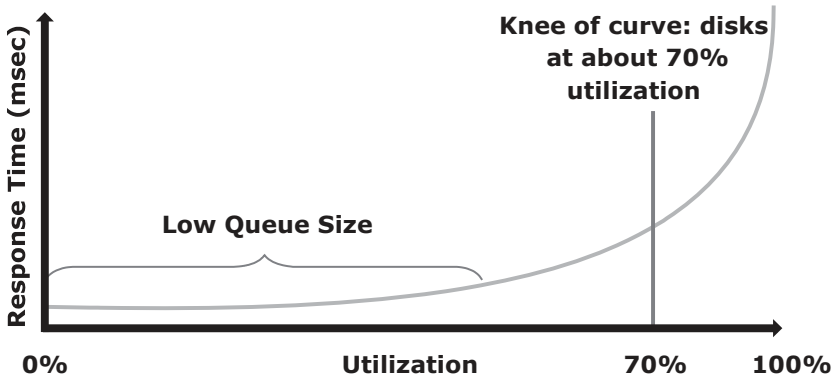


Figure 2-13: Utilization versus response time

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Chapter 4
Intelligent Storage Systems

KEY CONCEPTS

- Intelligent Storage Systems
- Cache Mirroring and Vaulting
- Logical Unit Number
- LUN Masking
- Meta LUN
- Virtual Storage Provisioning
- High-End Storage Systems
- Midrange Storage Systems

Business-critical applications require high levels of performance, availability, security, and scalability. A disk drive is a core element of storage that governs the performance of any storage system. Some of the older disk-array technologies could not overcome performance constraints due to the limitations of disk drives and their mechanical components. RAID technology made an important contribution to enhancing storage performance and reliability, but disk drives, even with a RAID implementation, could not meet the performance requirements of today's applications.

With advancements in technology, a new breed of storage solutions, known as *intelligent storage systems*, has evolved. These intelligent storage systems are feature-rich RAID arrays that provide highly optimized I/O processing capabilities. These storage systems are configured with a large amount of memory (called *cache*) and multiple I/O paths and use sophisticated algorithms to meet the requirements of performance-sensitive applications. These arrays have an operating environment that intelligently and optimally handles the management, allocation, and utilization of storage resources. Support for flash drives and other modern-day technologies, such as virtual storage provisioning and automated storage tiering, has added a new dimension to storage system performance, scalability, and availability.

This chapter covers components of intelligent storage systems along with storage provisioning to applications.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

4.1 Components of an Intelligent Storage System

An intelligent storage system consists of four key components: *front end*, *cache*, *back end*, and *physical disks*. Figure 4-1 illustrates these components and their interconnections. An I/O request received from the host at the front-end port is processed through cache and back end, to enable storage and retrieval of data from the physical disk. A read request can be serviced directly from cache if the requested data is found in the cache. In modern intelligent storage systems, front end, cache, and back end are typically integrated on a single board (referred to as a *storage processor* or *storage controller*).

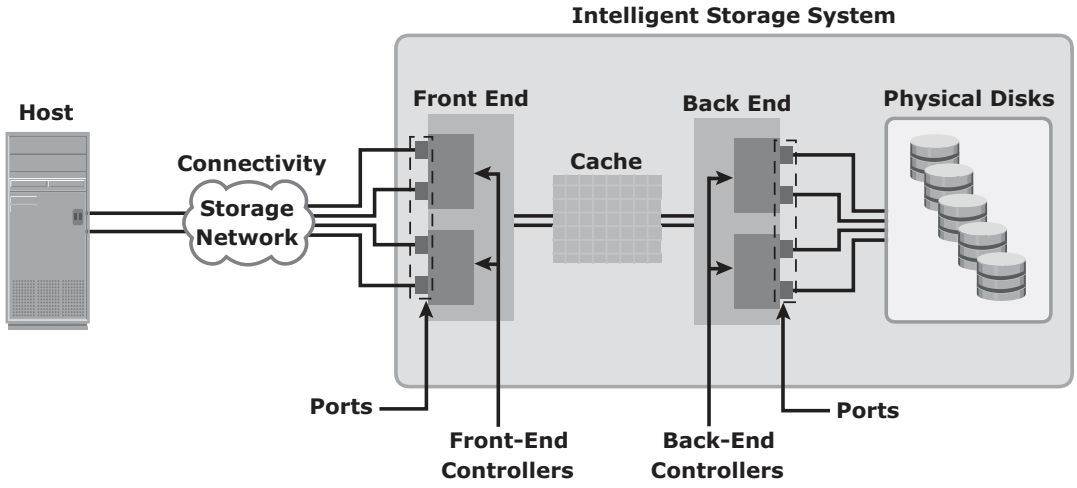


Figure 4-1: Components of an intelligent storage system

4.1.1 Front End

The front end provides the interface between the storage system and the host. It consists of two components: front-end ports and front-end controllers. Typically, a front end has redundant controllers for high availability, and each controller contains multiple ports that enable large numbers of hosts to connect to the intelligent storage system. Each front-end controller has processing logic that executes the appropriate transport protocol, such as Fibre Channel, iSCSI, FICON, or FCoE for storage connections.

Front-end controllers route data to and from cache via the internal data bus. When the cache receives the write data, the controller sends an acknowledgment message back to the host.

4.1.2 Cache

Cache is semiconductor memory where data is placed temporarily to reduce the time required to service I/O requests from the host.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Cache improves storage system performance by isolating hosts from the mechanical delays associated with rotating disks or hard disk drives (HDD). Rotating disks are the slowest component of an intelligent storage system. Data access on rotating disks usually takes several millisecond because of seek time and rotational latency. Accessing data from cache is fast and typically takes less than a millisecond. On intelligent arrays, write data is first placed in cache and then written to disk.

Structure of Cache

Cache is organized into pages, which is the smallest unit of cache allocation. The size of a cache page is configured according to the application I/O size. Cache consists of the *data store* and *tag RAM*. The data store holds the data whereas the tag RAM tracks the location of the data in the data store (see Figure 4-2) and in the disk.

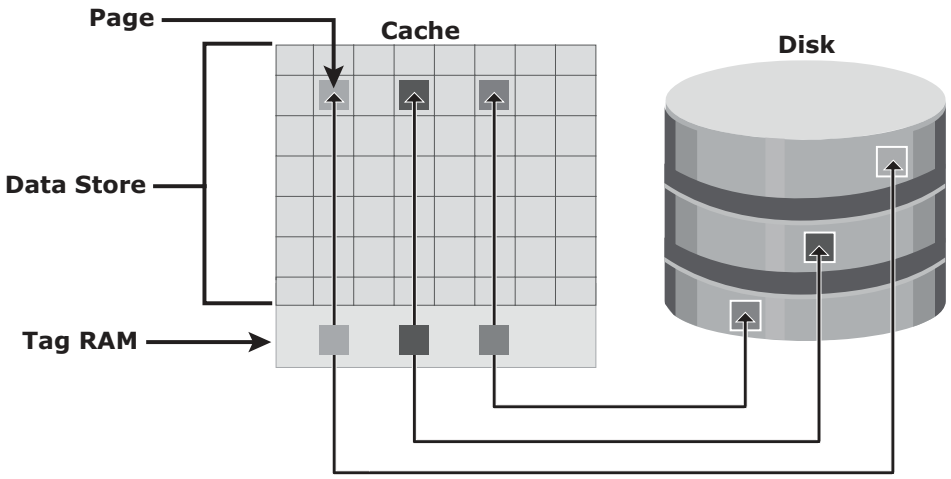


Figure 4-2: Structure of cache

Entries in tag RAM indicate where data is found in cache and where the data belongs on the disk. Tag RAM includes a *dirty bit* flag, which indicates whether the data in cache has been committed to the disk. It also contains time-based information, such as the time of last access, which is used to identify cached information that has not been accessed for a long period and may be freed up.

Read Operation with Cache

When a host issues a read request, the storage controller reads the tag RAM to determine whether the required data is available in cache. If the requested data is found in the cache, it is called a *read cache hit* or *read hit* and data is sent directly to the host, without any disk operation (see Figure 4-3 [a]). This provides

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

a fast response time to the host (about a millisecond). If the requested data is not found in cache, it is called a *cache miss* and the data must be read from the disk (see Figure 4-3 [b]). The back end accesses the appropriate disk and retrieves the requested data. Data is then placed in cache and finally sent to the host through the front end. Cache misses increase the I/O response time.

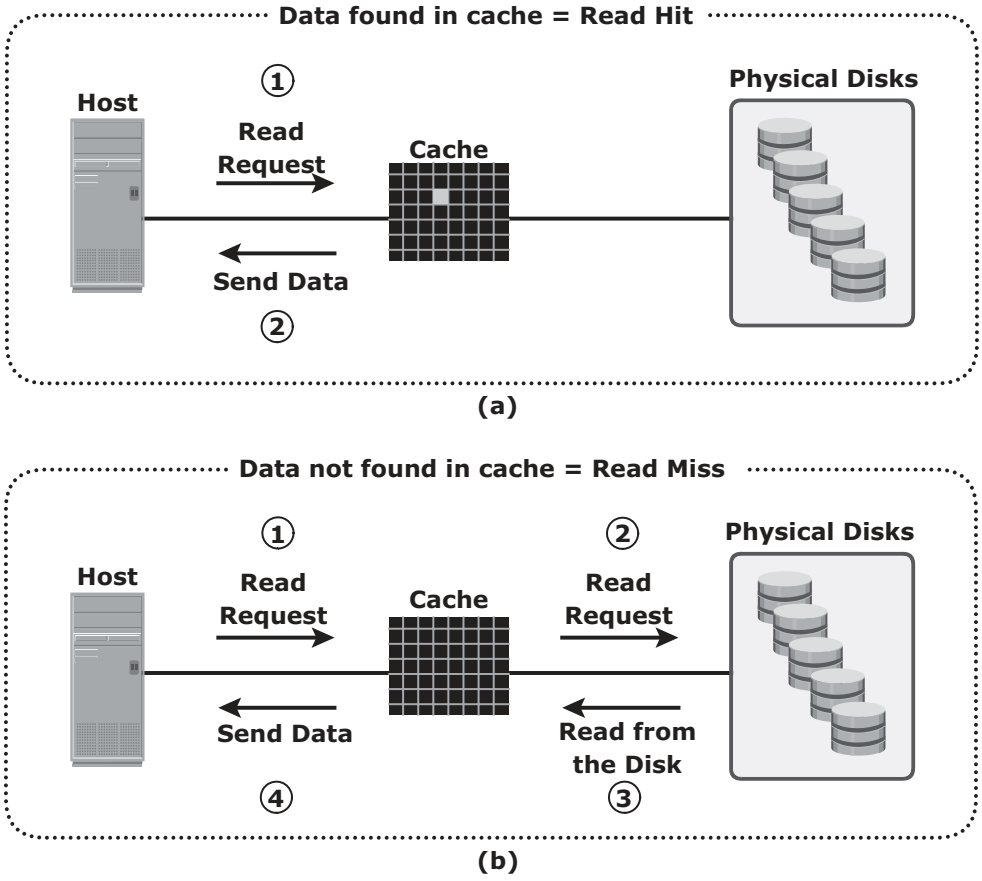


Figure 4-3: Read hit and read miss

A *prefetch* or *read-ahead* algorithm is used when read requests are sequential. In a sequential read request, a contiguous set of associated blocks is retrieved. Several other blocks that have not yet been requested by the host can be read from the disk and placed into cache in advance. When the host subsequently requests these blocks, the read operations will be read hits. This process significantly improves the response time experienced by the host. The intelligent storage system offers fixed and variable prefetch sizes. In *fixed prefetch*, the intelligent storage system prefetches a fixed amount of data. It is most suitable when host I/O sizes are uniform. In *variable prefetch*, the storage system prefetches an amount of data in multiples of the size of the host request. *Maximum prefetch* limits the number of data blocks that can be prefetched to prevent the disks from being rendered busy with prefetch at the expense of other I/Os.

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

Read performance is measured in terms of the *read hit ratio*, or the *hit rate*, usually expressed as a percentage. This ratio is the number of read hits with respect to the total number of read requests. A higher read hit ratio improves the read performance.

Write Operation with Cache

Write operations with cache provide performance advantages over writing directly to disks. When an I/O is written to cache and acknowledged, it is completed in far less time (from the host's perspective) than it would take to write directly to disk. Sequential writes also offer opportunities for optimization because many smaller writes can be coalesced for larger transfers to disk drives with the use of cache.

A write operation with cache is implemented in the following ways:

- **Write-back cache:** Data is placed in cache and an acknowledgment is sent to the host immediately. Later, data from several writes are committed (de-staged) to the disk. Write response times are much faster because the write operations are isolated from the mechanical delays of the disk. However, uncommitted data is at risk of loss if cache failures occur.
- **Write-through cache:** Data is placed in the cache and immediately written to the disk, and an acknowledgment is sent to the host. Because data is committed to disk as it arrives, the risks of data loss are low, but the write-response time is longer because of the disk operations.

Cache can be bypassed under certain conditions, such as large size write I/O. In this implementation, if the size of an I/O request exceeds the predefined size, called *write aside size*, writes are sent to the disk directly to reduce the impact of large writes consuming a large cache space. This is particularly useful in an environment where cache resources are constrained and cache is required for small random I/Os.

Cache Implementation

Cache can be implemented as either dedicated cache or global cache. With dedicated cache, separate sets of memory locations are reserved for reads and writes. In global cache, both reads and writes can use any of the available memory addresses. Cache management is more efficient in a global cache implementation because only one global set of addresses has to be managed.

Global cache allows users to specify the percentages of cache available for reads and writes for cache management. Typically, the read cache is small, but

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

it should be increased if the application being used is read-intensive. In other global cache implementations, the ratio of cache available for reads versus writes is dynamically adjusted based on the workloads.

Cache Management

Cache is a finite and expensive resource that needs proper management. Even though modern intelligent storage systems come with a large amount of cache, when all cache pages are filled, some pages have to be freed up to accommodate new data and avoid performance degradation. Various cache management algorithms are implemented in intelligent storage systems to proactively maintain a set of free pages and a list of pages that can be potentially freed up whenever required. The most commonly used algorithms are discussed in the following list:

- **Least Recently Used (LRU):** An algorithm that continuously monitors data access in cache and identifies the cache pages that have not been accessed for a long time. LRU either frees up these pages or marks them for reuse. This algorithm is based on the assumption that data that has not been accessed for a while will not be requested by the host. However, if a page contains write data that has not yet been committed to disk, the data is first written to disk before the page is reused.
- **Most Recently Used (MRU):** This algorithm is the opposite of LRU, where the pages that have been accessed most recently are freed up or marked for reuse. This algorithm is based on the assumption that recently accessed data may not be required for a while.

As cache fills, the storage system must take action to flush dirty pages (data written into the cache but not yet written to the disk) to manage space availability. *Flushing* is the process that commits data from cache to the disk. On the basis of the I/O access rate and pattern, high and low levels called *watermarks* are set in cache to manage the flushing process. *High watermark* (*HWM*) is the cache utilization level at which the storage system starts high-speed flushing of cache data. *Low watermark* (*LWM*) is the point at which the storage system stops flushing data to the disks. The cache utilization level, as shown in Figure 4-4, drives the mode of flushing to be used:

- **Idle flushing:** Occurs continuously, at a modest rate, when the cache utilization level is between the high and low watermark.
- **High watermark flushing:** Activated when cache utilization hits the high watermark. The storage system dedicates some additional resources for flushing. This type of flushing has some impact on I/O processing.
- **Forced flushing:** Occurs in the event of a large I/O burst when cache reaches 100 percent of its capacity, which significantly affects the I/O response time. In forced flushing, system flushes the cache on priority by allocating more resources.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

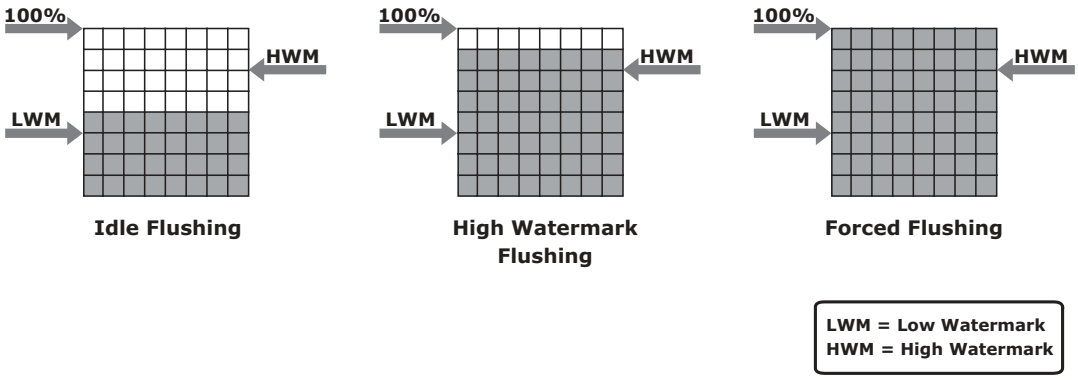


Figure 4-4: Types of flushing

Cache Data Protection

Cache is volatile memory, so a power failure or any kind of cache failure will cause loss of the data that is not yet committed to the disk. This risk of losing uncommitted data held in cache can be mitigated using *cache mirroring* and *cache vaulting*:

- **Cache mirroring:** Each write to cache is held in two different memory locations on two independent memory cards. If a cache failure occurs, the write data will still be safe in the mirrored location and can be committed to the disk. Reads are staged from the disk to the cache; therefore, if a cache failure occurs, the data can still be accessed from the disk. Because only writes are mirrored, this method results in better utilization of the available cache.

  In cache mirroring approaches, the problem of maintaining *cache coherency* is introduced. Cache coherency means that data in two different cache locations must be identical at all times. It is the responsibility of the array operating environment to ensure coherency.

- **Cache vaulting:** The risk of data loss due to power failure can be addressed in various ways: powering the memory with a battery until the AC power is restored or using battery power to write the cache content to the disk. If an extended power failure occurs, using batteries is not a viable option. This is because in intelligent storage systems, large amounts of data might need to be committed to numerous disks, and batteries might not provide power for sufficient time to write each piece of data to its intended disk. Therefore, storage vendors use a set of physical disks to dump the contents of cache during power failure. This is called *cache vaulting* and the disks are called *vault drives*. When power is restored, data from these disks is written back to write cache and then written to the intended disks.

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

SERVER FLASH-CACHING TECHNOLOGY

Server flash-caching technology uses intelligent caching software and a PCI Express (PCIe) flash card on the host. This dramatically improves application performance by reducing latency, and accelerates throughput. Server flash-caching technology works in both physical and virtual environments and provides performance acceleration for read-intensive workloads. This technology uses minimal CPU and memory resources from the server by offloading flash management onto the PCIe card.

It intelligently determines which data would benefit by sitting in the server on PCIe flash and closer to the application. This avoids the latencies associated with I/O access over the network to the storage array. With this, the processing power required for an application's most frequently referenced data is offloaded from the back-end storage to the PCIe card. Therefore, the storage array can allocate greater processing power to other applications.

4.1.3 Back End

The *back end* provides an interface between cache and the physical disks. It consists of two components: back-end ports and back-end controllers. The back-end controls data transfers between cache and the physical disks. From cache, data is sent to the back end and then routed to the destination disk. Physical disks are connected to ports on the back end. The back-end controller communicates with the disks when performing reads and writes and also provides additional, but limited, temporary data storage. The algorithms implemented on back-end controllers provide error detection and correction, along with RAID functionality.

For high data protection and high availability, storage systems are configured with dual controllers with multiple ports. Such configurations provide an alternative path to physical disks if a controller or port failure occurs. This reliability is further enhanced if the disks are also dual-ported. In that case, each disk port can connect to a separate controller. Multiple controllers also facilitate load balancing.

4.1.4 Physical Disk

Physical disks are connected to the back-end storage controller and provide persistent data storage. Modern intelligent storage systems provide support to a variety of disk drives with different speeds and types, such as FC, SATA, SAS, and flash drives. They also support the use of a mix of flash, FC, or SATA within the same array.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

4.2 Storage Provisioning

Storage provisioning is the process of assigning storage resources to hosts based on capacity, availability, and performance requirements of applications running on the hosts. Storage provisioning can be performed in two ways: traditional and virtual. *Virtual provisioning* leverages virtualization technology for provisioning storage for applications. This section details both traditional and virtual storage provisioning.

4.2.1 Traditional Storage Provisioning

In traditional storage provisioning, physical disks are logically grouped together and a required RAID level is applied to form a set, called a RAID set. The number of drives in the RAID set and the RAID level determine the availability, capacity, and performance of the RAID set. It is highly recommend that the RAID set be created from drives of the same type, speed, and capacity to ensure maximum usable capacity, reliability, and consistency in performance. For example, if drives of different capacities are mixed in a RAID set, the capacity of the smallest drive is used from each disk in the set to make up the RAID set's overall capacity. The remaining capacity of the larger drives remains unused. Likewise, mixing higher revolutions per minute (RPM) drives with lower RPM drives lowers the overall performance of the RAID set.

RAID sets usually have a large capacity because they combine the total capacity of individual drives in the set. *Logical units* are created from the RAID sets by partitioning (seen as slices of the RAID set) the available capacity into smaller units. These units are then assigned to the host based on their storage requirements.

Logical units are spread across all the physical disks that belong to that set. Each logical unit created from the RAID set is assigned a unique ID, called a *logical unit number* (LUN). LUNs hide the organization and composition of the RAID set from the hosts. LUNs created by traditional storage provisioning methods are also referred to as *thick LUNs* to distinguish them from the LUNs created by virtual provisioning methods.

Figure 4-5 shows a RAID set consisting of five disks that have been sliced, or partitioned, into two LUNs: LUN 0 and LUN 1. These LUNs are then assigned to Host1 and Host 2 for their storage requirements.

When a LUN is configured and assigned to a non-virtualized host, a bus scan is required to identify the LUN. This LUN appears as a raw disk to the operating system. To make this disk usable, it is formatted with a file system and then the file system is mounted.

In a virtualized host environment, the LUN is assigned to the hypervisor, which recognizes it as a raw disk. This disk is configured with the hypervisor file system, and then virtual disks are created on it. *Virtual disks* are files on the hypervisor

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

file system. The virtual disks are then assigned to virtual machines and appear as raw disks to them. To make the virtual disk usable to the virtual machine, similar steps are followed as in a non-virtualized environment. Here, the LUN space may be shared and accessed simultaneously by multiple virtual machines.

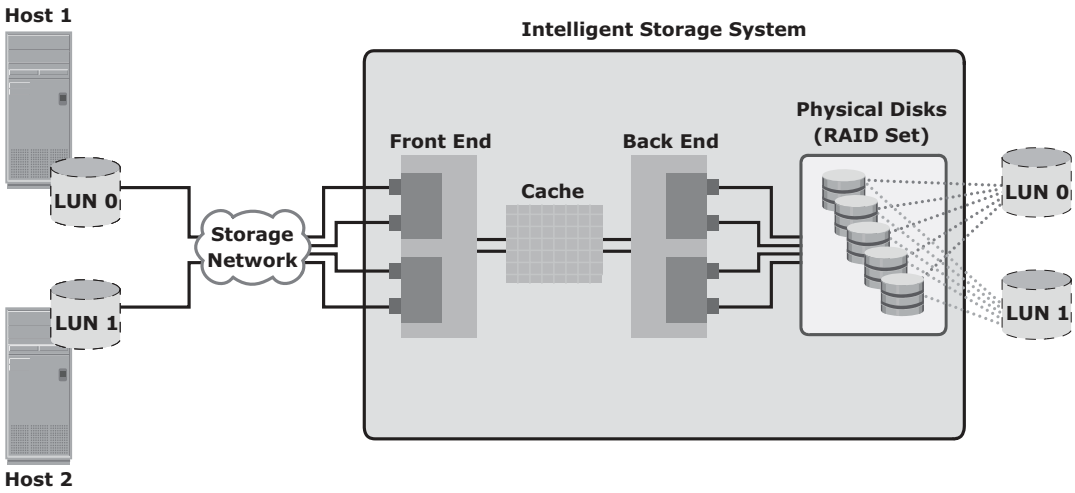


Figure 4-5: RAID set and LUNs

Virtual machines can also access a LUN directly on the storage system. In this method the entire LUN is allocated to a single virtual machine. Storing data in this way is recommended when the applications running on the virtual machine are response-time sensitive, and sharing storage with other virtual machines may impact their response time. The direct access method is also used when a virtual machine is clustered with a physical machine. In this case, the virtual machine is required to access the LUN that is being accessed by the physical machine.

LUN Expansion: MetaLUN

MetaLUN is a method to expand LUNs that require additional capacity or performance. A metaLUN can be created by combining two or more LUNs. A metaLUN consists of a base LUN and one or more component LUNs. MetaLUNs can be either *concatenated* or *striped*.

Concatenated expansion simply adds additional capacity to the base LUN. In this expansion, the component LUNs are not required to be of the same capacity as the base LUN. All LUNs in a concatenated metaLUN must be either protected (parity or mirrored) or unprotected (RAID 0). RAID types within a metaLUN can be mixed. For example, a RAID 1/0 LUN can be concatenated with a RAID 5 LUN. However, a RAID 0 LUN can be concatenated only with another RAID 0 LUN.

Concatenated expansion is quick but does not provide any performance benefit. (See Figure 4-6.)

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

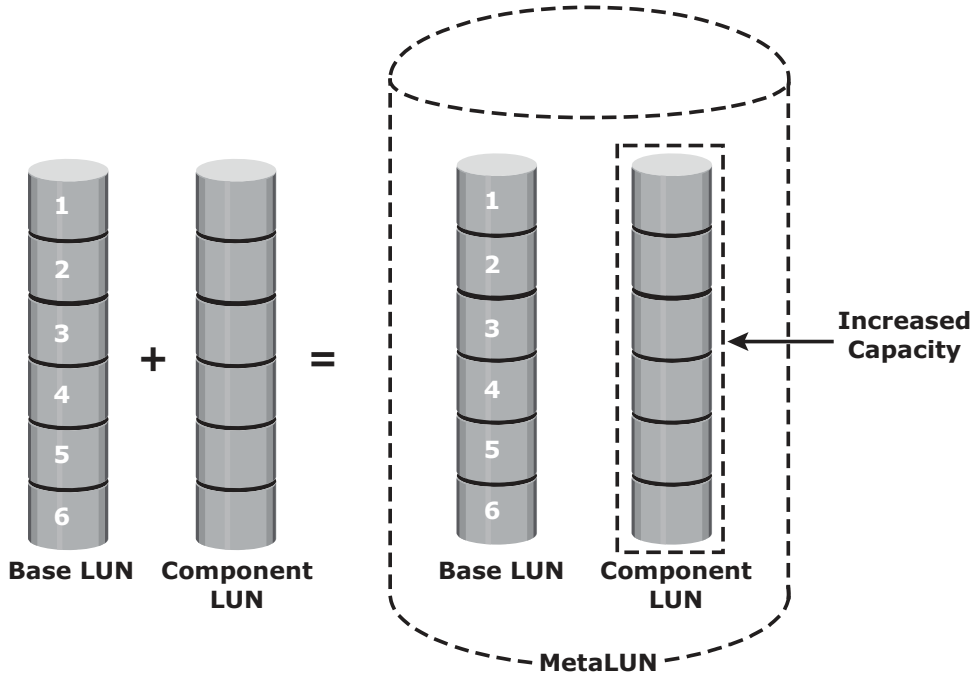


Figure 4-6: Concatenated metaLUN

Striped expansion restripes the base LUN's data across the base LUN and component LUNs. In striped expansion, all LUNs must be of the same capacity and RAID level. Striped expansion provides improved performance due to the increased number of drives being striped (see Figure 4-7).

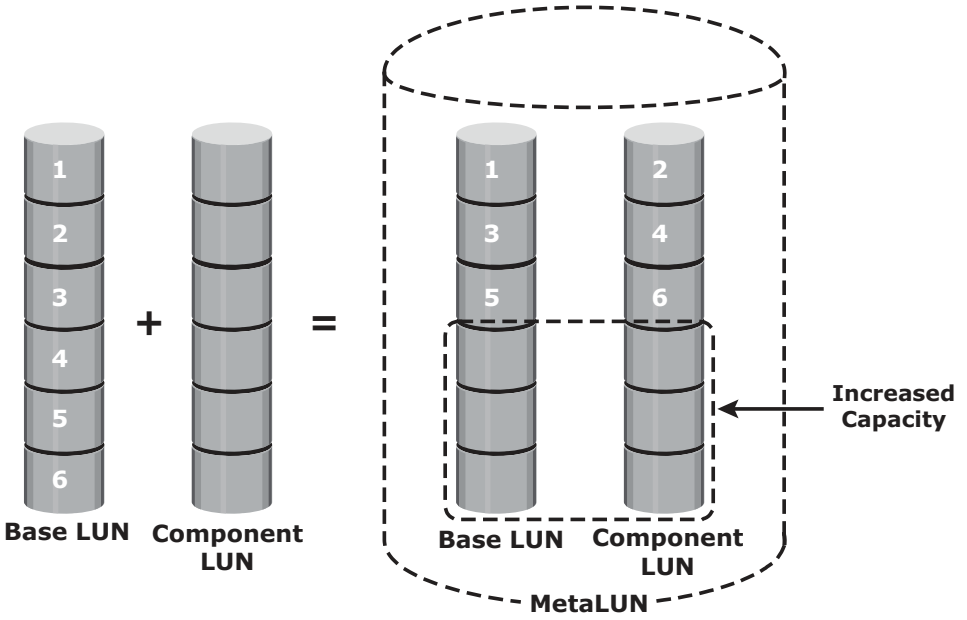


Figure 4-7: Striped metaLUN

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

All LUNs in both concatenated and striped expansion must reside on the same disk-drive type: either all Fibre Channel or all ATA.

4.2.2 Virtual Storage Provisioning

Virtual provisioning enables creating and presenting a LUN with more capacity than is physically allocated to it on the storage array. The LUN created using virtual provisioning is called a *thin LUN* to distinguish it from the traditional LUN.

Thin LUNs do not require physical storage to be completely allocated to them at the time they are created and presented to a host. Physical storage is allocated to the host "on-demand" from a shared pool of physical capacity. A *shared pool* consists of physical disks. A shared pool in virtual provisioning is analogous to a RAID group, which is a collection of drives on which LUNs are created. Similar to a RAID group, a shared pool supports a single RAID protection level. However, unlike a RAID group, a shared pool might contain large numbers of drives. Shared pools can be homogeneous (containing a single drive type) or heterogeneous (containing mixed drive types, such as flash, FC, SAS, and SATA drives).

Virtual provisioning enables more efficient allocation of storage to hosts. Virtual provisioning also enables oversubscription, where more capacity is presented to the hosts than is actually available on the storage array. Both shared pool and thin LUN can be expanded nondisruptively as the storage requirements of the hosts grow. Multiple shared pools can be created within a storage array, and a shared pool may be shared by multiple thin LUNs. Figure 4-8 illustrates the provisioning of thin LUNs.

Comparison between Virtual and Traditional Storage Provisioning

Administrators typically allocate storage capacity based on anticipated storage requirements. This generally results in the over provisioning of storage capacity, which then leads to higher costs and lower capacity utilization. Administrators often over-provision storage to an application for various reasons, such as, to avoid frequent provisioning of storage if the LUN capacity is exhausted, and to reduce disruption to application availability. Over provisioning of storage often leads to additional storage acquisition and operational costs.

Virtual provisioning addresses these challenges. Virtual provisioning improves storage capacity utilization and simplifies storage management. Figure 4-9 shows an example, comparing virtual provisioning with traditional storage provisioning.

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

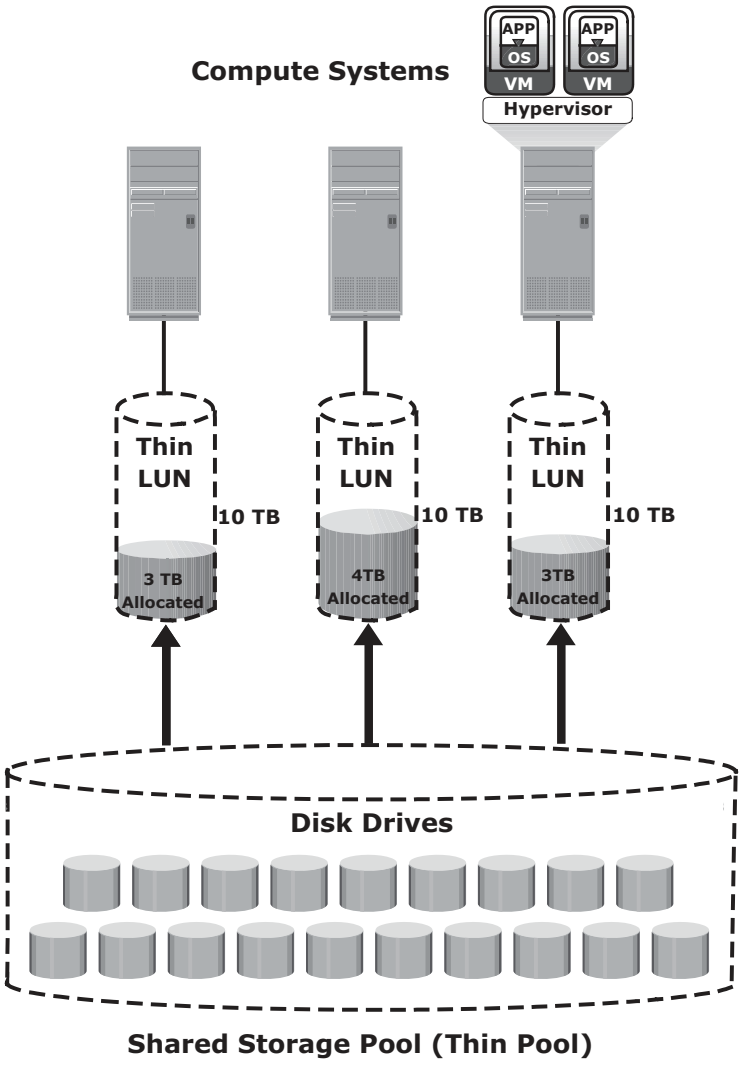


Figure 4-8: Virtual provisioning

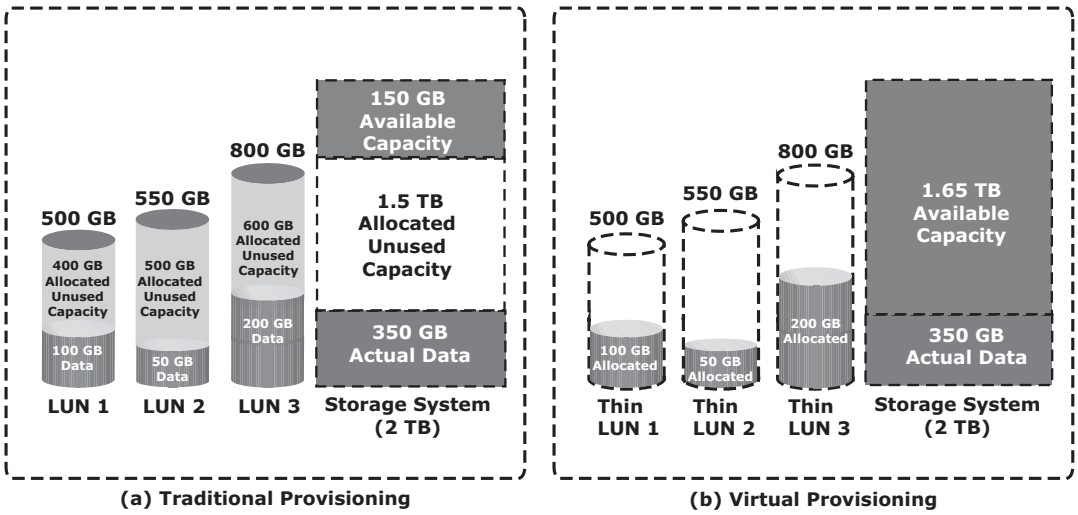


Figure 4-9: Traditional versus virtual provisioning

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

With traditional provisioning, three LUNs are created and presented to one or more hosts (see Figure 4-9 [a]). The total storage capacity of the storage system is 2 TB. The allocated capacity of LUN 1 is 500 GB, of which only 100 GB is consumed, and the remaining 400 GB is unused. The size of LUN 2 is 550 GB, of which 50 GB is consumed, and 500 GB is unused. The size of LUN 3 is 800 GB, of which 200 GB is consumed, and 600 GB is unused. In total, the storage system has 350 GB of data, 1.5 TB of allocated but unused capacity, and only 150 GB of remaining capacity available for other applications.

Now consider the same 2 TB storage system with virtual provisioning (see Figure 4-9 [b]). Here, three thin LUNs of the same sizes are created. However, there is no allocated unused capacity. In total, the storage system with virtual provisioning has the same 350 GB of data, but 1.65 TB of capacity is available for other applications, whereas only 150 GB is available in traditional storage provisioning.

Use Cases for Thin and Traditional LUNs

Virtual provisioning and thin LUN offer many benefits, although in some cases traditional LUN is better suited for an application. Thin LUNs are appropriate for applications that can tolerate performance variations. In some cases, performance improvement is perceived when using a thin LUN, due to striping across a large number of drives in the pool. However, when multiple thin LUNs contend for shared storage resources in a given pool, and when utilization reaches higher levels, the performance can degrade. Thin LUNs provide the best storage space efficiency and are suitable for applications where space consumption is difficult to forecast. Using thin LUNs benefits organizations in reducing power and acquisition costs and in simplifying their storage management.

Traditional LUNs are suited for applications that require predictable performance. Traditional LUNs provide full control for precise data placement and allow an administrator to create LUNs on different RAID groups if there is any workload contention. Organizations that are not highly concerned about storage space efficiency may still use traditional LUNs.

Both traditional and thin LUNs can coexist in the same storage array. Based on the requirement, an administrator may migrate data between thin and traditional LUNs.

4.2.3 LUN Masking

LUN masking is a process that provides data access control by defining which LUNs a host can access. The LUN masking function is implemented on the storage array. This ensures that volume access by hosts is controlled appropriately, preventing unauthorized or accidental use in a shared environment.

For example, consider a storage array with two LUNs that store data of the sales and finance departments. Without LUN masking, both departments

This document is available free of charge on studocu

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

can easily see and modify each other's data, posing a high risk to data integrity and security. With LUN masking, LUNs are accessible only to the designated hosts.

4.3 Types of Intelligent Storage Systems

Intelligent storage systems generally fall into one of the following two categories:

- High-end storage systems
- Midrange storage systems

Traditionally, high-end storage systems have been implemented with *active-active configuration*, whereas midrange storage systems have been implemented with *active-passive configuration*. The distinctions between these two implementations are becoming increasingly insignificant.

4.3.1 High-End Storage Systems

High-end storage systems, referred to as *active-active arrays*, are generally aimed at large enterprise applications. These systems are designed with a large number of controllers and cache memory. An active-active array implies that the host can perform I/Os to its LUNs through any of the available controllers (see Figure 4-10).

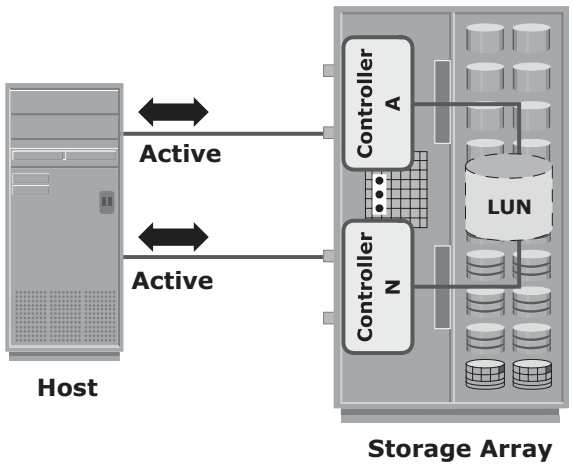


Figure 4-10: Active-active configuration

To address enterprise storage needs, these arrays provide the following capabilities:

- Large storage capacity
- Large amounts of cache to service host I/Os optimally

Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)

- Fault tolerance architecture to improve data availability
- Connectivity to mainframe computers and open systems hosts
- Availability of multiple front-end ports and interface protocols to serve a large number of hosts
- Availability of multiple back-end controllers to manage disk processing
- Scalability to support increased connectivity, performance, and storage capacity requirements
- Ability to handle large amounts of concurrent I/Os from a number of hosts and applications
- Support for array-based local and remote data replication

In addition to these features, high-end systems possess some unique features that are required for mission-critical applications.

4.3.2 Midrange Storage Systems

Midrange storage systems are also referred to as *active-passive arrays* and are best suited for small- and medium-sized enterprise applications. They also provide optimal storage solutions at a lower cost. In an active-passive array, a host can perform I/Os to a LUN only through the controller that owns the LUN. As shown in Figure 4-11, the host can perform reads or writes to the LUN only through the path to controller A because controller A is the owner of that LUN. The path to controller B remains passive and no I/O activity is performed through this path.

Midrange storage systems are typically designed with two controllers, each of which contains host interfaces, cache, RAID controllers, and interface to disk drives.

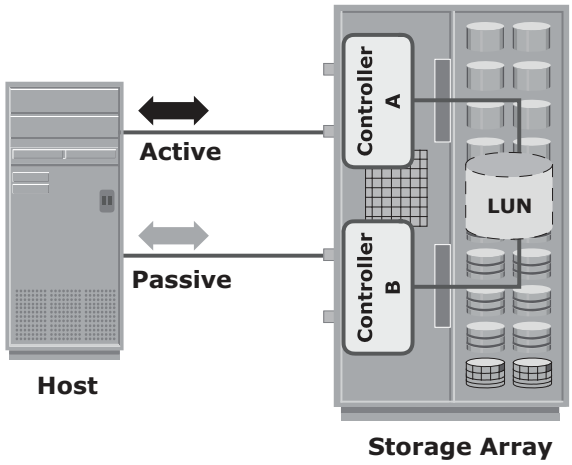


Figure 4-11: Active-passive configuration

This document is available free of charge on studocu
Downloaded by S SRI KRISHNA (s.sri.krishna.050202@gmail.com)