Data centred environment

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

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

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

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.

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

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.

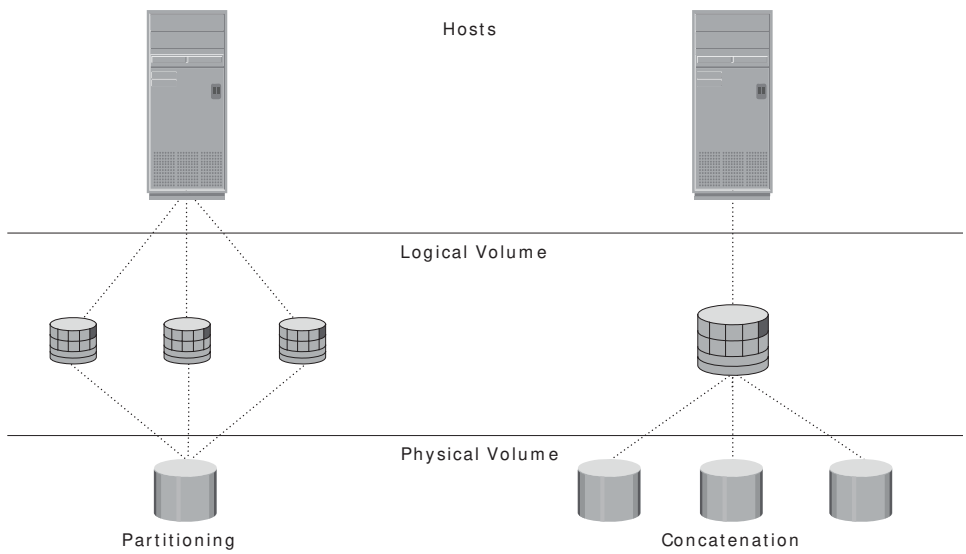


Figure 2-1: Disk partitioning and concatenation

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:

- 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.

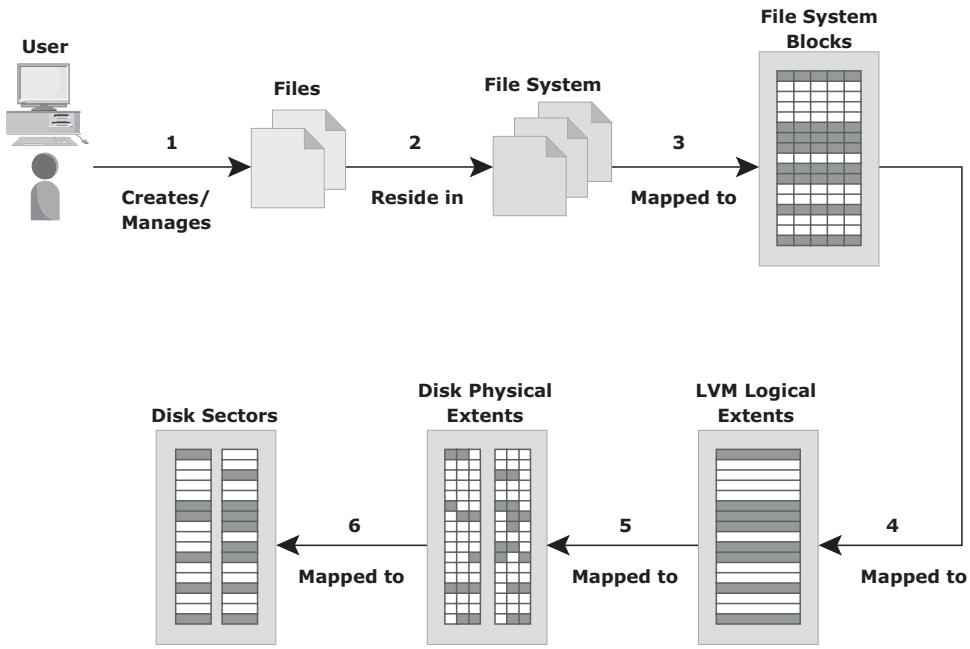
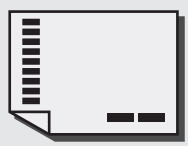


Figure 2-2: Process of mapping user files to disk storage

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

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.

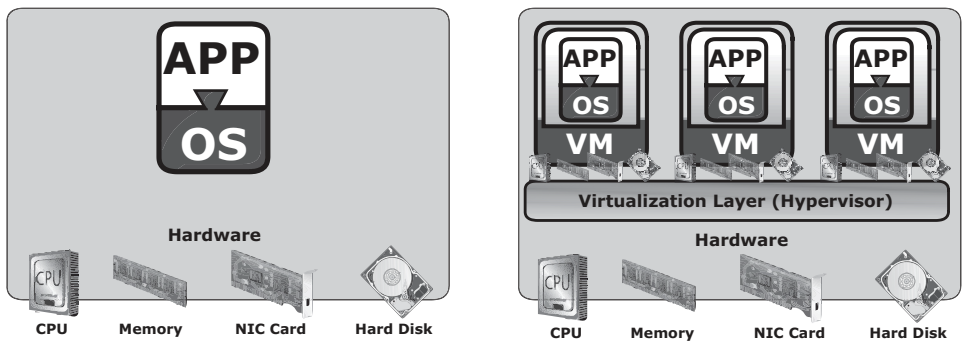
Dedicated file servers may be installed to manage and share a large number of files over a network. These file servers support multiple file systems and use file-sharing protocols specific to the operating system — for example, NFS and CIFS. These protocols are detailed in Chapter 7.

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.

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.

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

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

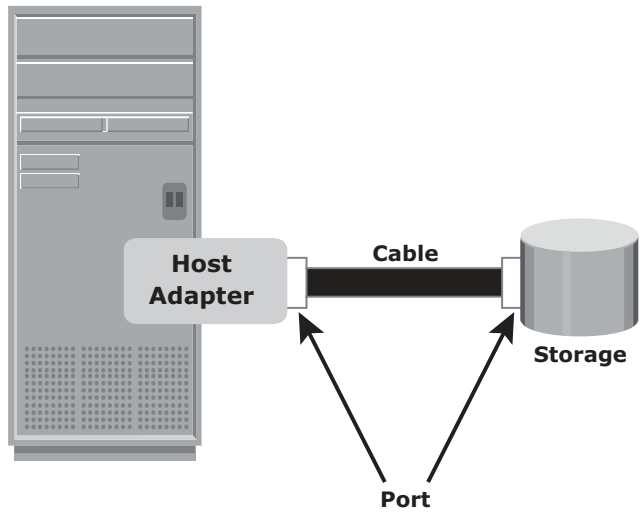


Figure 2-4: Physical components of connectivity

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.

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

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.

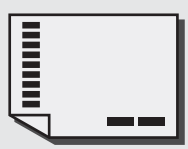
- In a shared computing environment, data stored on tape cannot be accessed by multiple applications simultaneously, restricting its use to one application at a time.
- On a tape drive, the read/write head touches the tape surface, so the tape degrades or wears out after repeated use.
- The storage and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

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.

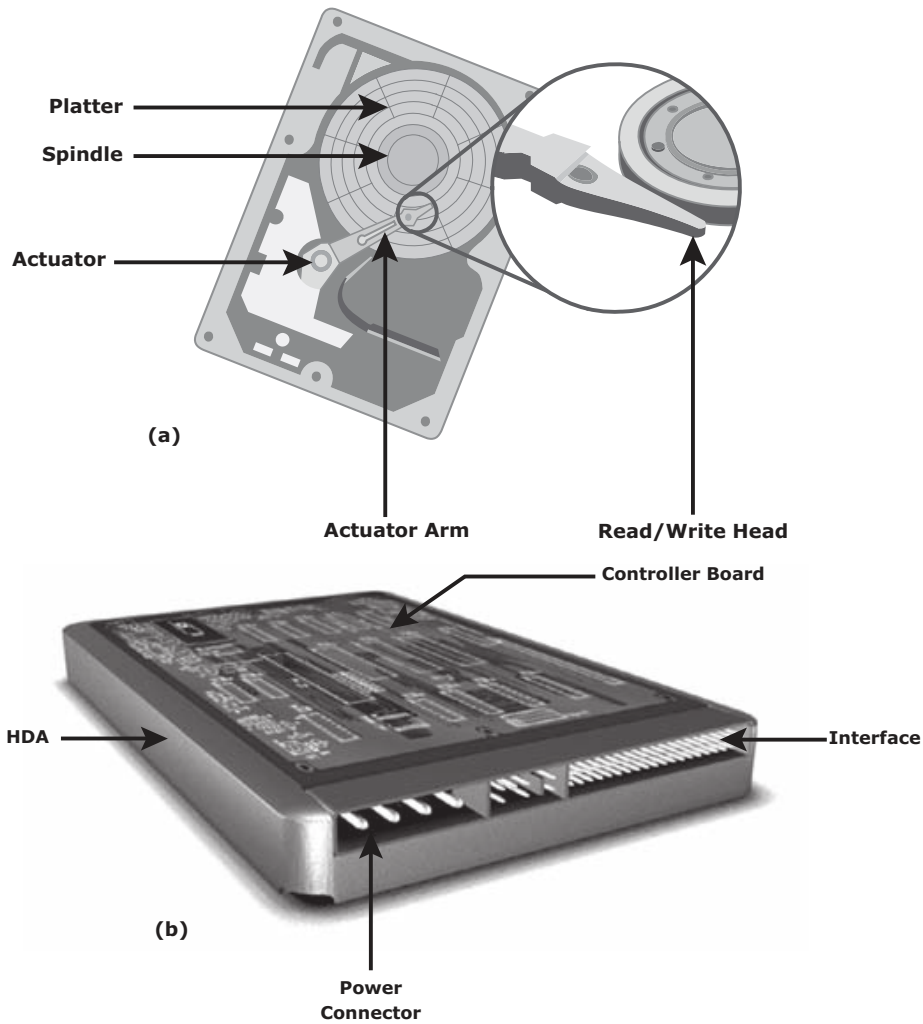


Figure 2-5: Disk drive components

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.

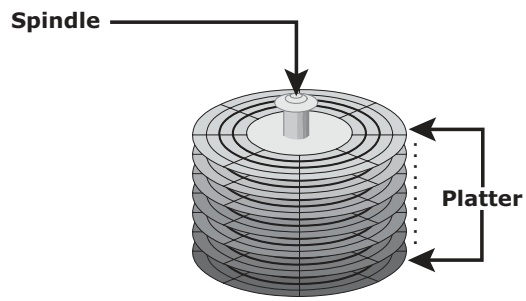


Figure 2-6: Spindle and platter

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*

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

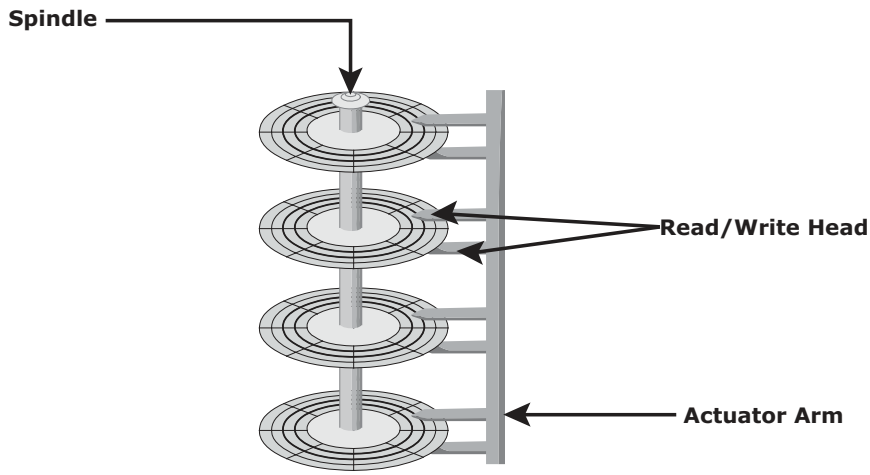


Figure 2-7: Actuator arm assembly

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.

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.

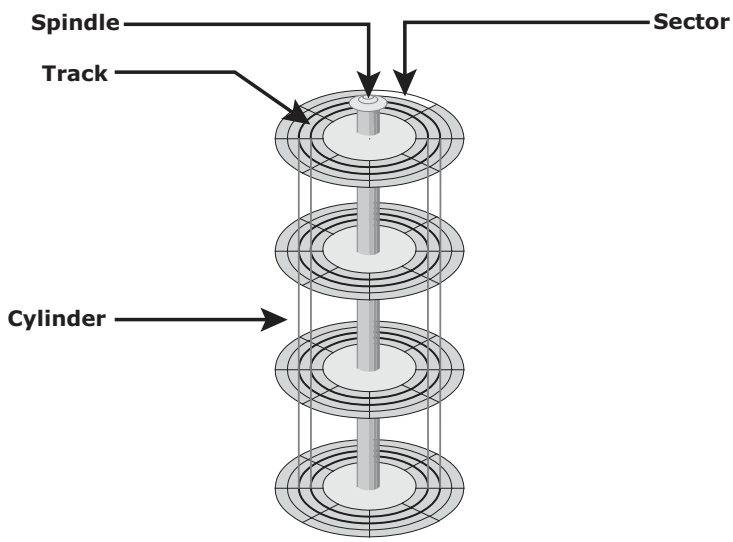


Figure 2-8: Disk structure: sectors, tracks, and cylinders

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.

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.

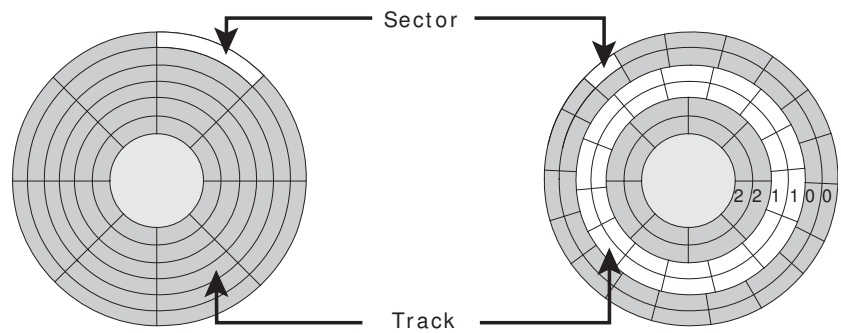
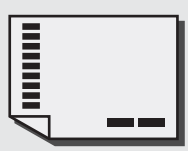


Figure 2-9: Zoned bit recording



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.

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.

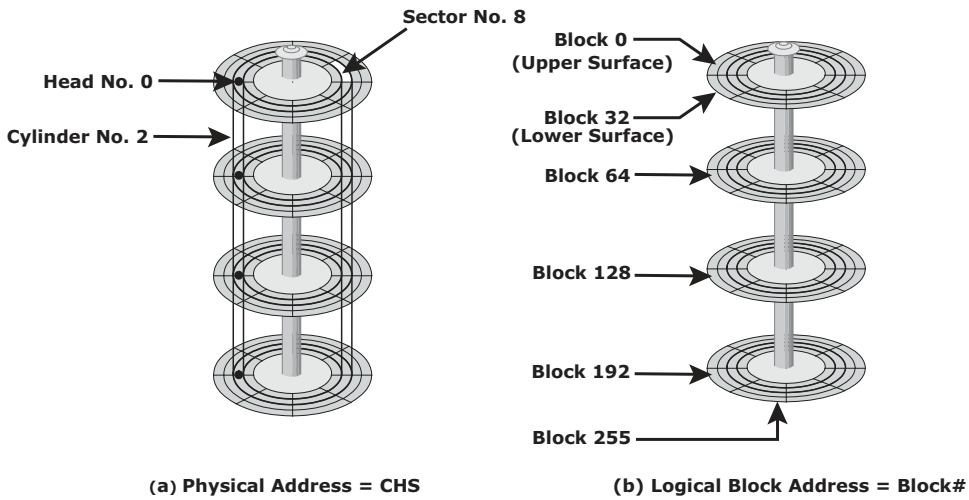


Figure 2-10: Physical address and logical block address

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.

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,

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.

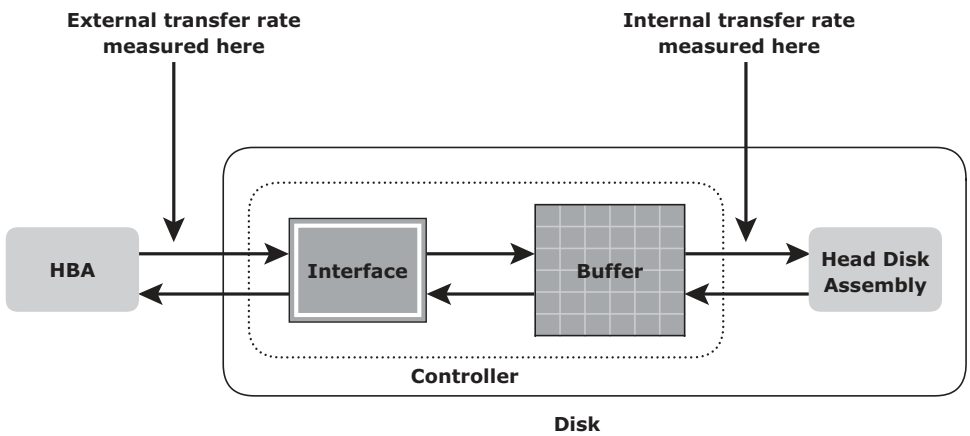


Figure 2-11: Data transfer rate

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.

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.

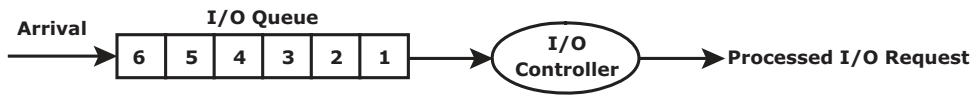


Figure 2-12: I/O processing

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.

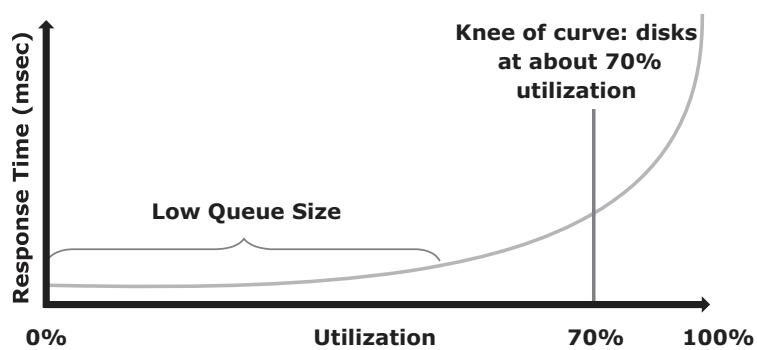


Figure 2-13: Utilization versus response time

The graph indicates that the response time changes are nonlinear as the utilization increases. When the average queue sizes are low, the response time remains low. The response time increases slowly with added load on the queue and increases exponentially when the utilization exceeds 70 percent. Therefore, for performance-sensitive applications, it is common to utilize disks below their 70 percent of I/O serving capability.

2.8 Host Access to Data

Data is accessed and stored by applications using the underlying infrastructure. The key components of this infrastructure are the operating system (or file system), connectivity, and storage. The storage device can be internal and (or) external to the host. In either case, the host controller card accesses the storage devices using predefined protocols, such as IDE/ATA, SCSI, or Fibre Channel (FC). IDE/ATA and SCSI are popularly used in small and personal computing environments for accessing internal storage. FC and iSCSI protocols are used for accessing data from an external storage device (or subsystems). External storage devices can be connected to the host directly or through the storage network. When the storage is connected directly to the host, it is referred as *direct-attached storage* (DAS), which is detailed later in this chapter.

Understanding access to data over a network is important because it lays the foundation for storage networking technologies. Data can be accessed over a network in one of the following ways: block level, file level, or object level.

In general, the application requests data from the file system (or operating system) by specifying the filename and location. The file system maps the file attributes to the logical block address of the data and sends the request to the storage device. The storage device converts the logical block address (LBA) to a cylinder-head-sector (CHS) address and fetches the data.

In a block-level access, the file system is created on a host, and data is accessed on a network at the block level, as shown in Figure 2-14 (a). In this case, raw disks or logical volumes are assigned to the host for creating the file system.

In a file-level access, the file system is created on a separate file server or at the storage side, and the file-level request is sent over a network, as shown in Figure 2-14 (b). Because data is accessed at the file level, this method has higher overhead, as compared to the data accessed at the block level. Object-level access is an intelligent evolution, whereby data is accessed over a network in terms of self-contained objects with a unique object identifier. Details of storage networking technologies and deployments are covered in Section II of this book, "Storage Networking Technologies."