STORAGE SYSTEM

1.2Evolution of Storage Architecture

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

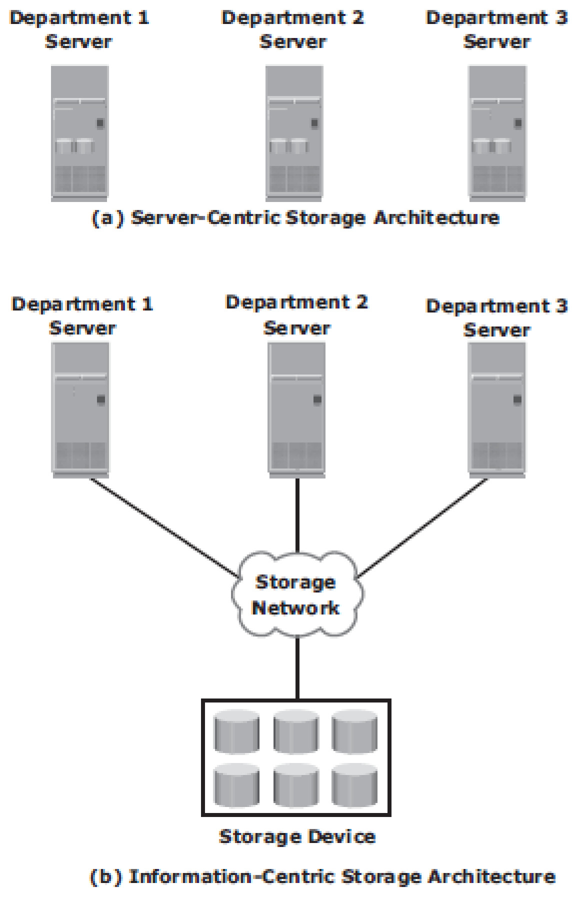


Figure 1-4: Evolution of storage architecture

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 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.

1.2 Datacenter Infrastructure

- Organizations maintain data centers to provide centralized data processing capabilities across the enterprise.
- Data centers store and manage large amounts of mission-critical data.
- The data center infrastructure includes computers, storage systems, network devices, dedicated power backups, and environmental controls (such as air conditioning and fire suppression).
- Large organizations often maintain more than one data center to distribute data processing workloads and provide backups in the event of a disaster.
- The storage requirements of a data center are met by a combination of various storage architectures

Core Elements

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

➢ **Application:** An application is a computer program that provides the logic for computing operations. Applications, such as an order processing system, can be layered on a database, which in turn uses operating system services to perform read/write operations to storage devices

➢ **Database:** More commonly, a database management system (DBMS) provides a structured way to store data in logically organized tables that are interrelated. A DBMS optimizes the storage and retrieval of data.

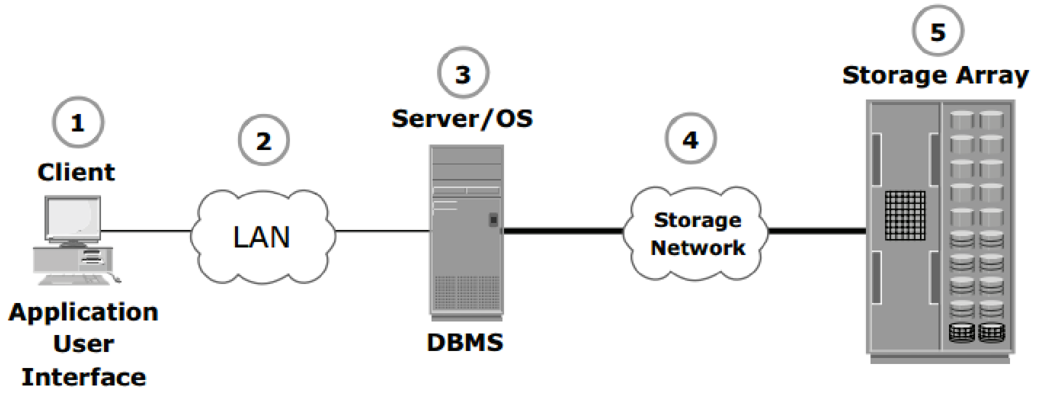
➢ **Server and operating system:** A computing platform (hardware, software and firmware) that runs applications and databases.

➢ **Network:** A data path that facilitates communication between clients and servers or between servers and storage.

➢ **Storage array:** 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.

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



1. A customer places an order through the AUI of the order processing application software located on the client computer.
2. The client connects to the server over the LAN and accesses the DBMS located on the server to update the relevant information such as the customer name, address, payment method, products ordered, and quantity ordered.
3. The DBMS uses the server operating system to read and write this data to the database located on physical disks in the storage array.
4. The Storage Network provides the communication link between the server and the storage array and transports the read or write commands between them.
5. The storage array, after receiving the read or write commands from the server, performs the necessary operations to store the data on physical disks.

Figure 1-5: Example of an order processing system

Key Requirements for Data Center Elements

- Uninterrupted operation of datacenters is critical to the survival and success of a business.
- It is necessary to have a reliable infrastructure that ensures data is accessible at all times.
- While the requirements, shown in Figure 1-6, are applicable to all elements of the data center infrastructure, our focus here is on storage systems.

➢ **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:** Polices, procedures, and proper integration of the data center core elements that will prevent unauthorized access to information must be established. In addition to the security measures for client access, specific mechanisms must enable servers to access only their allocated resources on storage arrays.

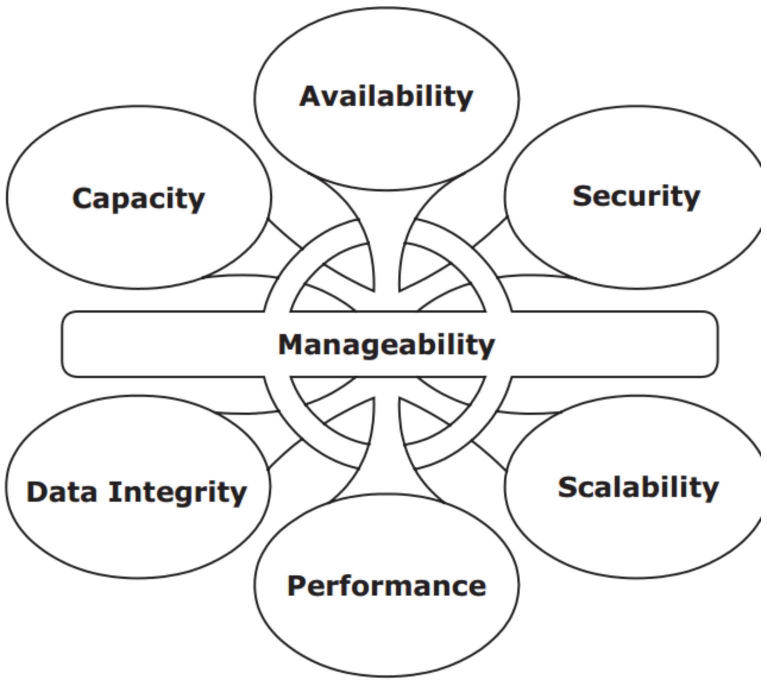


Figure 1-6: Key characteristics of data center elements

- **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 written to disk 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 be able to provide additional capacity without interrupting availability, or, at the very least, with minimal disruption. Capacity may be managed by reallocation of existing resources, rather than by adding new resources.
- **Manageability:** A data center should perform all operations and activities in the most efficient manner. Manageability can be achieved through automation and the reduction of human (manual) intervention in common tasks.

1.2.3 Managing Storage Infrastructure

- Managing a modern, complex data center involves many tasks.
- Key management activities include:

- **Monitoring** is the continuous collection of information and the review of the entire data center infrastructure. The aspects of a data center that are monitored include security, performance, accessibility, and capacity.
- **Reporting** 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** is the process of providing the hardware, software, and other resources needed to run a data center. Provisioning activities include capacity and resource planning. Capacity planning ensures that the user's and the application's future needs will be addressed in the most cost-effective and controlled manner. Resource planning is the process of evaluating and identifying required resources, such as personnel, the facility (site), and the technology. Resource planning ensures that adequate resources are available to meet user and application requirements.

- For example, the utilization of an application's allocated storage capacity may be monitored.
- As soon as utilization of the storage capacity reaches a critical value, additional storage capacity may be provisioned to the application.
- If utilization of the storage capacity is properly monitored and reported, business growth can be understood and future capacity requirements can be anticipated.
- This helps to frame a proactive data management policy

1.3VIRTUALIZATION:

- Virtualization is a technique of abstracting physical resources, such as compute, storage and network , and making them appear as logical resources.
- Virtualization enables pooling of resources and providing an aggregation view of the physical resource capabilities.
- It also enables centralized management of pooled resources.
- Virtual resources can be created and provisioned from the pooled physical resources.

CLOUD COMPUTING:

- Cloud computing enables individuals or businesess 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.
- It empowers self-service requesting through a fully automated request-fulfillment process.
- It enables consumption –based metering
- Cloud infrastructure is usually built upon virtualized data centres.

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.

1.4Application:

- 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.

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.

- Application virtualization eliminates this conflict among multiple applications by isolating different versions of an application and the associated O/S resources.

1.5 Host

- 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.
- 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.

1.5.1 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 fi le pages is slower than access to physical memory pages because swap fi le pages are allocated on the disk drive, which is slower than physical memory.

1.5.2 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.

1.5.3 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 fi le 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 fi le 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 fi le 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.

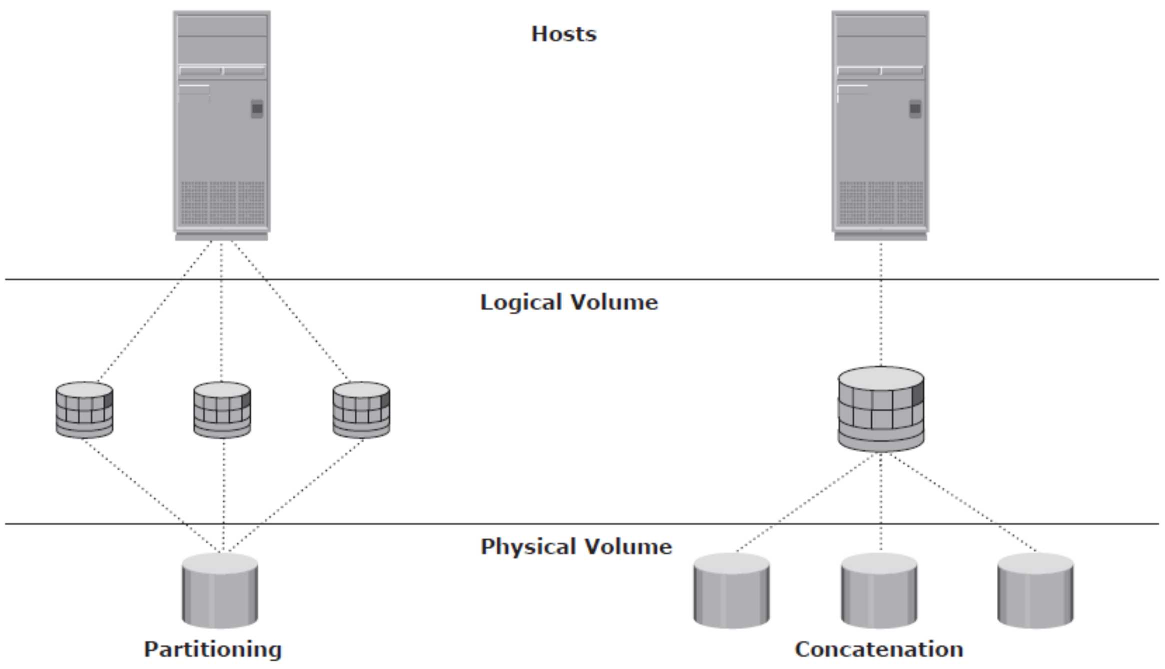


Figure 2-1: Disk partitioning and concatenation

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 fi le 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.

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.

1.5.4 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 fi les. A file system enables easy access to data fi les 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 fi le 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 fi le 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 fi le system is fixed at the time of its creation.

The file system size depends on the block size and the total number of fi le 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 fi le system becomes increasingly fragmented.

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.

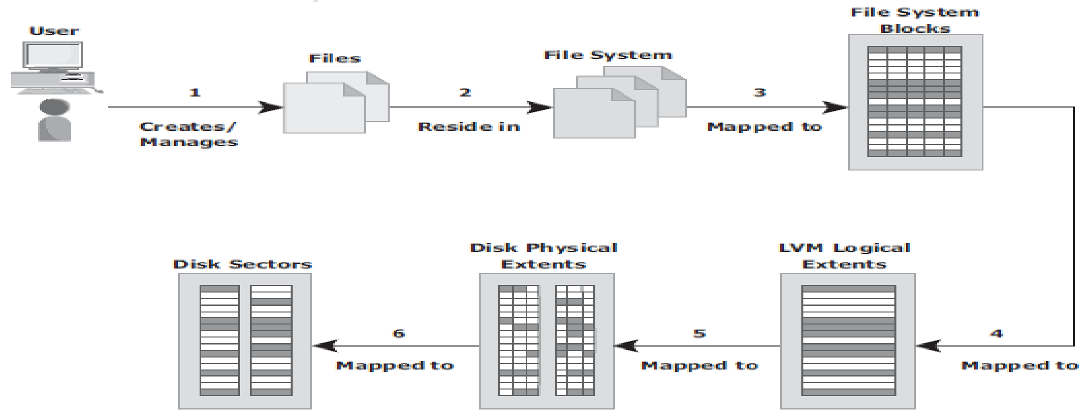


Figure 2-2: Process of mapping user files to disk storage

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

1. Files are created and managed by users and applications.
2. These fi les reside in the fi le systems.
3. The fi le systems are mapped to fi le system blocks.

4. The fi le 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.

A file system can be either a **journaling** file system or a **nonjournaling** fi le 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 fi le system attempts to update the metadata structures by examining and repairing them. This operation takes a long time on large fi le systems. If there is insufficient information to re-create the wanted or original structure, the fi les might be misplaced or lost, resulting in corrupted fi le 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 fi le system, they are written to this separate area. After the journal has been updated, the operation on the fi le 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 fi le system. In addition, because information about the pending operation is saved, the risk of files being lost is reduced.

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

1.5.5 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 fi les 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.

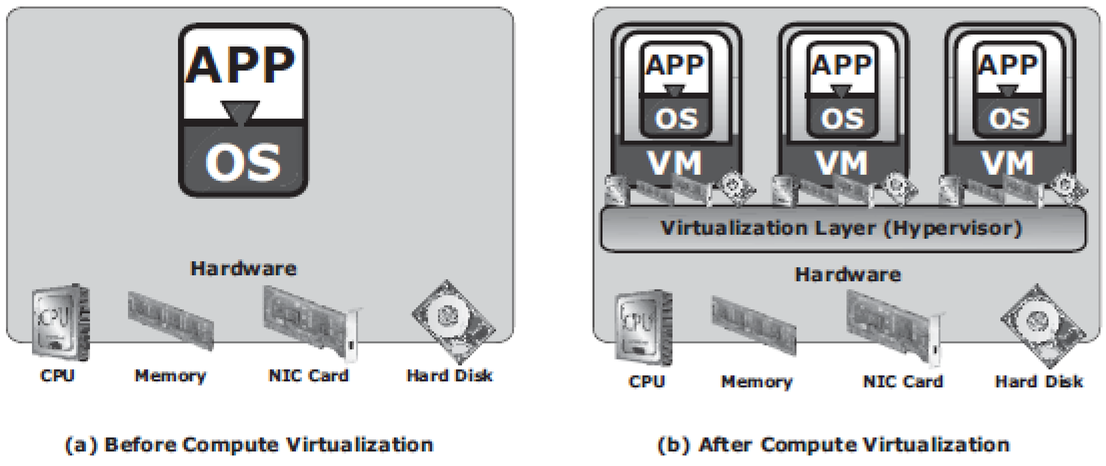

APP
OS
Hardware
CPU
Memory
NIC Card
Hard Disk
(a) Before Compute Virtualization
APP
OS
VM
Virtualization Layer (Hypervisor)
Hardware
CPU
Memory
NIC Card
Hard Disk
(b) After Compute Virtualization

Figure 2-3: Server virtualization

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.

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.

1.6Connectivity

- Connectivity refers to the interconnection between hosts or between a host and any other peripheral devices, such as printers or storage devices.
- The discussion here focuses 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.

1.6.1Physical 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.

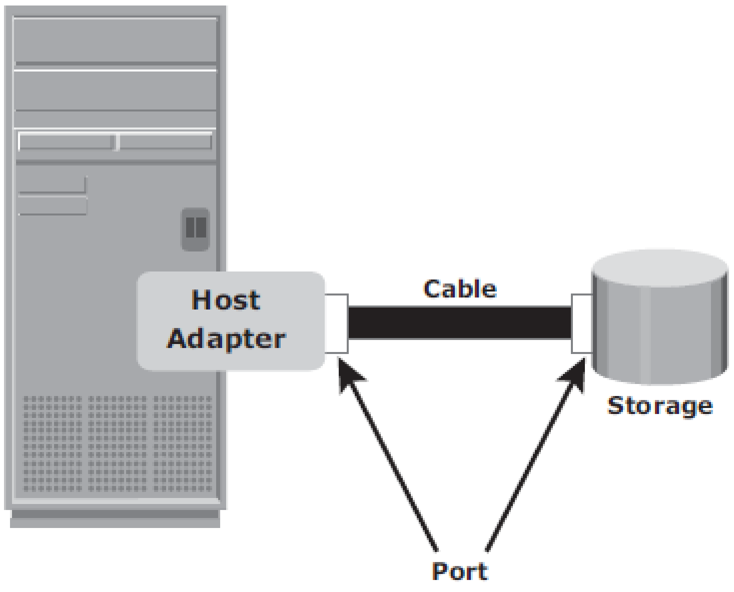


Figure 2-4: Physical components of connectivity

1.6.2 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.

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.

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.

2.5 Storage

- The storage device is the most important component in the storage system environment.
- A storage device uses magnetic or solid state media.
- Disks, tapes, and diskettes use magnetic media.
- CD/DVD is an example of a storage device that uses optical media, and removable flash memory card is an example of solid state media.
- Tapes are a popular storage media used for backup because of their relatively low cost.

- In the past, data centers hosted a large number of tape drives and processed several thousand reels of tape.
- However, tape has the following limitations:
  - Data is stored on the tape linearly along the length of the tape. Search and retrieval of data is done sequentially, invariably taking 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 tape and the overhead associated with managing tape media are significant.
- In spite of its limitations, tape is widely deployed for its cost effectiveness and mobility. Continued development of tape technology is resulting in high capacity Medias and high speed drives.
- Modern tape libraries come with additional memory (cache) and / or disk drives to increase data throughput. With these and added intelligence, today's tapes are part of an end-to-end data management solution, especially as a low-cost solution for storing infrequently accessed data and as long-term data storage.

Optical disk 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/laptop computers. It is also used as a distribution medium for single applications, such as games, or as a means of transferring small amounts of data from one self-contained system to another.

Optical disks have limited capacity and speed, which limits 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 disk storage. A CD-ROM is an example of a WORM device.

Optical disks, to some degree, guarantee that the content has not been altered, so they can be used as low-cost alternatives for long-term storage of relatively small amounts of fixed content that will not change after it is created.

Collections of optical disks in an array, called jukeboxes, are still used as a fixed-content storage solution. Other forms of optical disks include CD-RW and 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.

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.

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.

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.

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.
- 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.

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 andthe 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 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 an 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 fi le or swap fi le) 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 fi le 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 fi le so that enough RAM is available for processes that are more active.
- Access to swap fi le 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.

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.

Volume Manager:

- The evolution of Logical Volume Managers (LVMs) enabled dynamic extension of fi le 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.

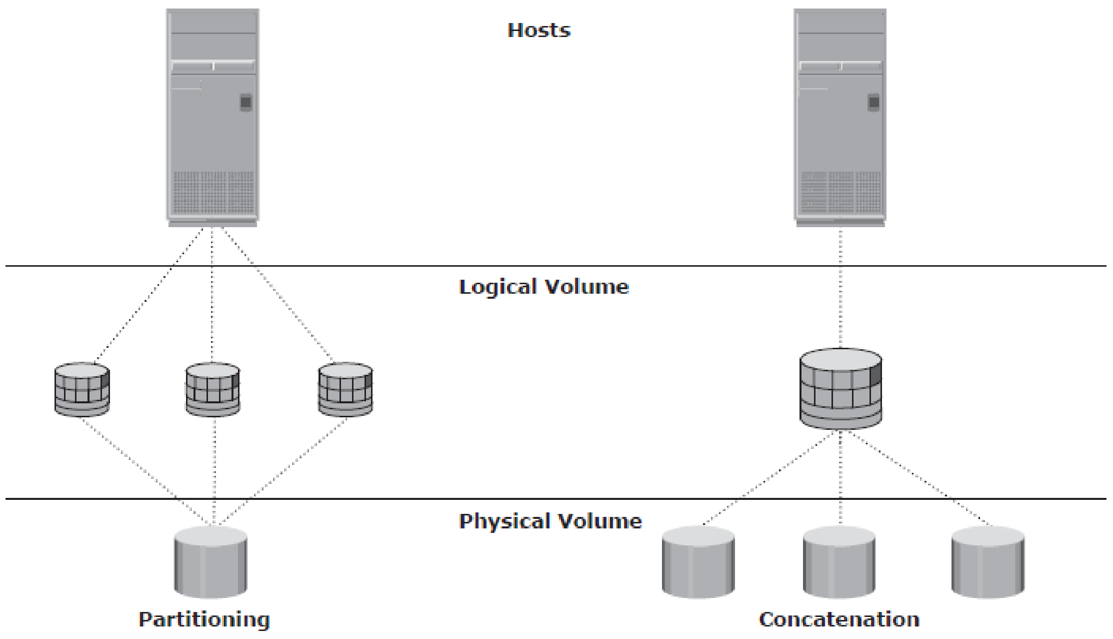


Figure 2-1: Disk partitioning and concatenation

- 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 simplifi es 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 fi le 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.

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.
- File systems enable easy access to data files residing within a disk drive, a disk partition, or a logical volume.
- A file system needs host-based 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 the files on the disks is controlled by the permissions given 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.
- Some of the common file systems are as follows:
    - FAT 32 (File Allocation Table) for Microsoft Windows
    - NT File System (NTFS) for Microsoft Windows
    - UNIX File System (UFS) for UNIX

- Extended File System (EXT 2/3) 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, 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 has to be consistent in order for the file system to be considered healthy.
- A upper block contains important information about the file system, such as the file system type, creation and modification dates, size and layout, the count of available resources (such as number of free blocks, inodes, etc.), and a flag indicating the mount status of the file system.
- An inode is associated with every file and directory and contains information about file length, ownership, access privileges, time of last access/modification, number of links, and the addresses for finding the location on the physical disk where the actual data is stored.

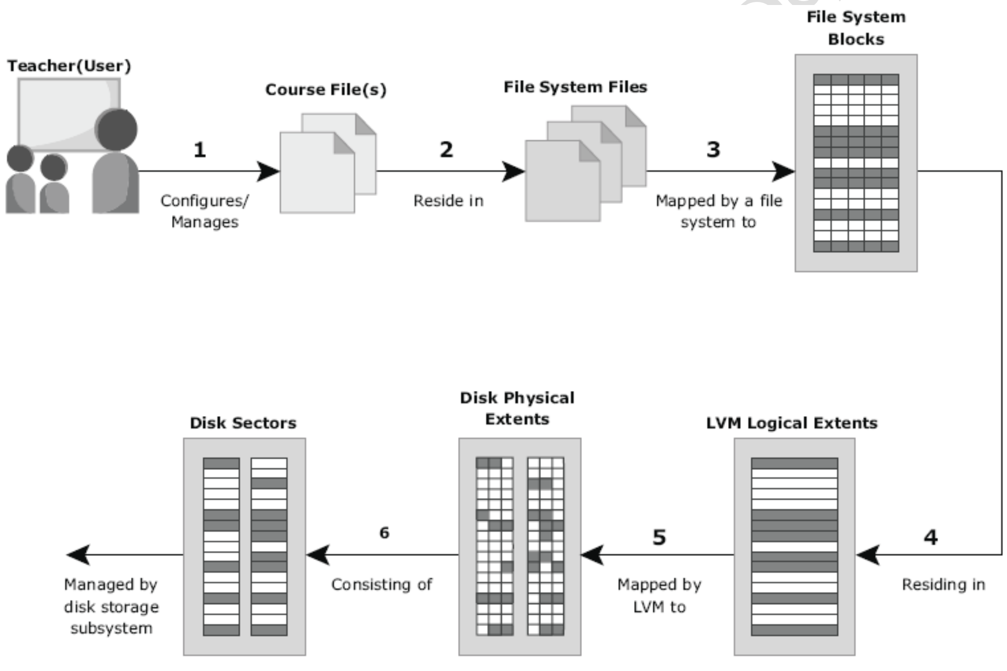


Figure 2-12: Process of mapping user files to disk storage

- A file system block is the smallest "container" of physical disk space allocated for 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.
- File system size depends on block size and the total number of blocks of data stored.
- 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 (i.e., become fragmented) when new blocks are added or deleted.
- Over time, as files grow larger, the file system becomes increasingly fragmented.

- Figure 2-12 shows the following process of mapping user files to the disk storage subsystem with an LVM:

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

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

Non-Journaling File System Vs Journaling 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 fi le system attempts to update the metadata structures by examining and repairing them.
- This operation takes a long time on large fi le 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 fi le 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.
- However, the much shortened time for fi le system checks and the file system integrity provided by journaling far outweighs its disadvantage.
- Nearly all file system implementations today use journaling.

COMPUTE VIRTUALIZATION

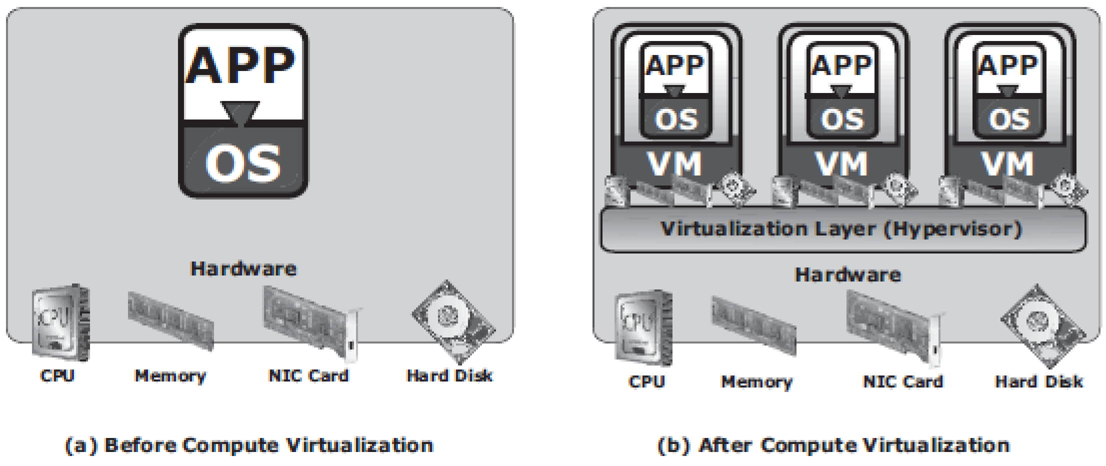

APP
OS
Hardware
CPU
Memory
NIC Card
Hard Disk
(a) Before Compute Virtualization
APP
OS
VM
Virtualization Layer (Hypervisor)
Hardware
CPU
Memory
NIC Card
Hard Disk
(b) After Compute Virtualization

Figure 2-3: Server 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.
- 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.

Storage

- The storage device is the most important component in the storage system environment.
- A storage device uses magnetic or solid state media.
- Disks, tapes, and diskettes use magnetic media.
- CD-ROM is an example of a storage device that uses optical media, and removable flash memory card is an example of solid state media.

Tapes are a popular storage media used for backup because of their relatively low cost.
In the past, data centers hosted a large number of tape drives and processed several thousand reels of tape.
However, tape has the following limitations:

Data is stored on the tape linearly along the length of the tape. Search and retrieval of data is done sequentially, invariably taking 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 tape and the overhead associated with managing tape media are significant.
In spite of its limitations, tape is widely deployed for its cost effectiveness and mobility.
Continued development of tape technology is resulting in high capacity Medias and high speed drives.

Modern tape libraries come with additional memory (cache) and / or disk drives to increase data throughput. With these and added intelligence, today’s tapes are part of an end-to-end data management solution, especially as a low-cost solution for storing infrequently accessed data and as long-term data storage.

Optical disk 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/laptop computers.

It is also used as a distribution medium for single applications, such as games, or as a means of transferring small amounts of data from one self-contained system to another.

Optical disks have limited capacity and speed, which limits 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 disk storage.

A CD-ROM is an example of a WORM device. Optical disks, to some degree, guarantee that the content has not been altered, so they can be used as low-cost alternatives for long-term storage of relatively small amounts of fixed content that will not change after it is created.

Collections of optical disks in an array, called jukeboxes, are still used as a fixed content storage solution. Other forms of optical disks include CD-RW and 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.

Connectivity:

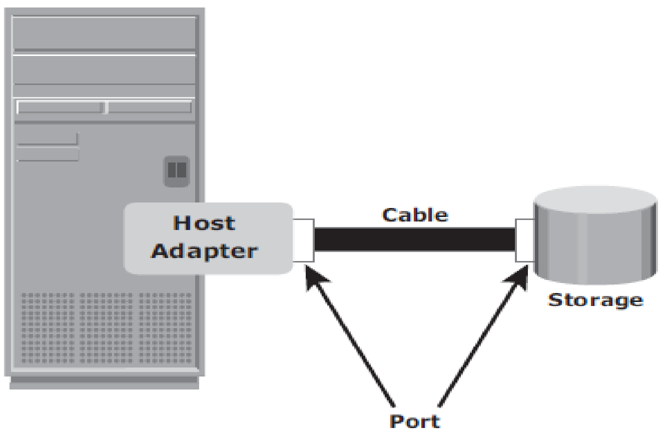


Figure 2-4: Physical components of connectivity

Interface Protocols: 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.

Interface Protocols: 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.

Interface Protocols: 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.

Interface Protocols: 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.

RAID

In the late 1980s, rapid adoption of computers for business processes stimulated the growth of new applications and databases, significantly increasing the demand for storage capacity. At that time, data was stored on a single large, expensive disk drive called Single Large Expensive Drive (SLED).

Use of single disks could not meet the required performance levels, due to their inherent limitations HDDs are susceptible to failures due to mechanical wear and tear and other environmental factors. An HDD failure may result in data loss.

The solutions available during the 1980s were not able to meet the availability and performance demands of applications. An HDD has a projected life expectancy before it fails. Mean Time Between Failure (MTBF) measures (in hours) the average life expectancy of an HDD.

Today, data centers deploy thousands of HDDs in their storage infrastructures. The greater the number of HDDs in a storage array, the greater the probability of a disk failure in the array.

For example, consider a storage array of 100 HDDs, each with an MTBF of 750,000 hours. The MTBF of this collection of HDDs in the array, therefore, is 750,000/100 or 7,500 hours. This means that a HDD in this array is likely to fail at least once in 7,500 hours.

RAID is an enabling technology that leverages multiple disks as part of a set, which provides data protection against HDD failures.

In general, RAID implementations also improve the I/O performance of storage systems by storing data across multiple HDDs

In 1987, Patterson, Gibson, and Katz at the University of California, Berkeley, published a paper titled “A Case for Redundant Arrays of Inexpensive Disks (RAID).” This paper described the use of small-capacity, inexpensive disk drives as an alternative to large-capacity drives common on mainframe computers.

The term RAID has been redefined to refer to independent disks, to reflect advances in the storage technology. RAID storage has now grown from an academic concept to an industry standard.

RAID Implementation Methods

There are two types of RAID implementation, hardware and software. Both have their merits and demerits and are discussed in this section.

Software RAID

- Software RAID uses host-based software to provide RAID functions.
- It is implemented at the operating-system level and does not use a dedicated hardware controller to manage the RAID array.
- Software RAID implementations offer cost and simplicity benefits when compared with hardware RAID.
- However, they have the following limitations:

Performance:

- Software RAID affects overall system performance.
- This is due to the additional CPU cycles required to perform RAID calculations.
- The performance impact is more pronounced for complex implementations of RAID.

Supported features:

- Software RAID does not support all RAID levels.

Operating system compatibility:

- Software RAID is tied to the host operating system hence upgrades to software RAID or to the operating system should be validated for compatibility.
- This leads to inflexibility in the data processing environment.

Hardware RAID

- In hardware RAID implementations, a specialized hardware controller is implemented either on the host or on the array.
- These implementations vary in the way the storage array interacts with the host.
- Controller card RAID is host-based hardware RAID implementation in which a specialized RAID controller is installed in the host and HDDs are connected to it.
- The RAID Controller interacts with the hard disks using a PCI bus.
- Manufacturers also integrate RAID controllers on motherboards.
- This integration reduces the overall cost of the system, but does not provide the flexibility required for high-end storage systems.
- The external RAID controller is an array-based hardware RAID.
- It acts as an interface between the host and disks.
- It presents storage volumes to the host, which manage the drives using the supported protocol.
- Key functions of RAID controllers are:
  - ➢ Management and control of disk aggregations
  - ➢ Translation of I/O requests between logical disks and physical disks
  - ➢ Data regeneration in the event of disk failures

RAID Techniques:

RAID techniques — striping, mirroring, and parity — form the basis for defining various RAID levels.

1 Striping:

- A RAID set is a group of disks.
- Within each disk, a predefined number of contiguously addressable disk blocks are defined as strips.
- The set of aligned strips that spans across all the disks within the RAID set is called a stripe.
- Figure 3-2 shows physical and logical representations of a striped RAID set.

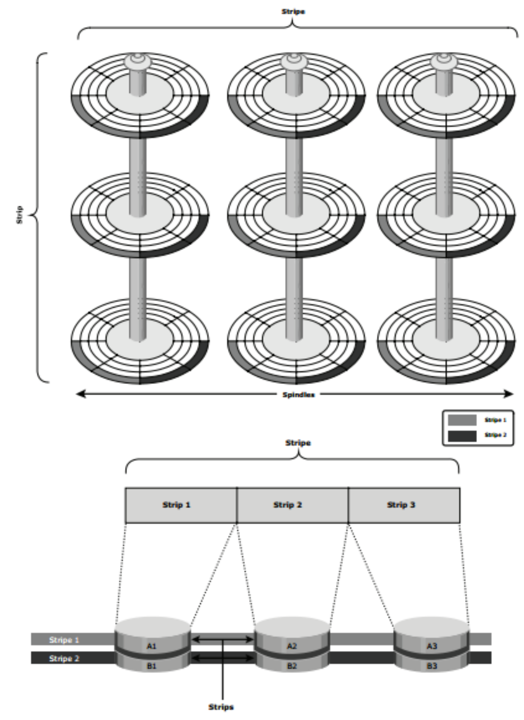


Figure 3-2: Striped RAID set

- Strip size(also called stripe depth) describes the number of blocks in a strip, and is the maximum amount of data that can be written to or read from a single HDD in the set before the next HDD is accessed, assuming that the accessed data starts at the beginning of the strip.
- Note that all strips in a stripe have the same number of blocks, and decreasing strip size means that data is broken into smaller pieces when spread across the disks.
- Stripe size is a multiple of strip size by the number of HDDs in the RAID set.
- Stripe width refers to the number of data strips in a stripe.
- Striped RAID does not protect data unless parity or mirroring is used.
- However, striping may significantly improve I/O performance.
- Depending on the type of RAID implementation, the RAID controller can be configured to access data across multiple HDDs simultaneously.

2 Mirroring:

- Mirroring is a technique whereby data is stored on two different HDDs, yielding two copies of data.
- In the event of one HDD failure, the data is intact on the surviving HDD (see Figure 3-3) and the controller continues to service the host's data requests from the surviving disk of a mirrored pair.

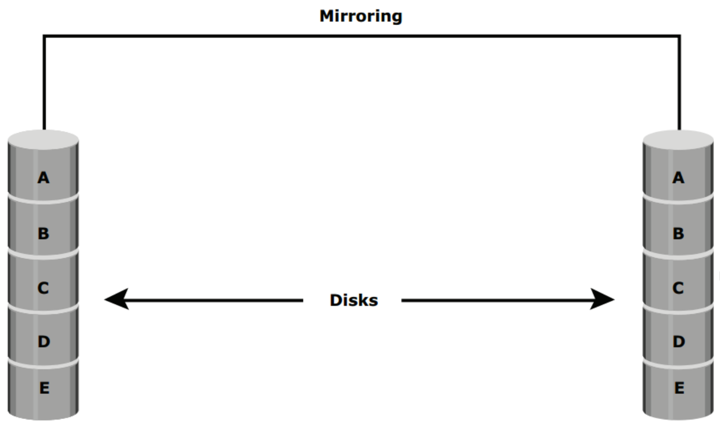


Figure 3-3: Mirrored disks in an array

- When the failed disk is replaced with a new disk, the controller copies the data from the surviving disk of the mirrored pair.
- This activity is transparent to the host.
- In addition to providing complete data redundancy, mirroring enables faster recovery from disk failure.
- However, disk mirroring provides only data protection and is not a substitute for data backup.
- Mirroring constantly captures changes in the data, whereas a backup captures point-in-time images of data.
- Mirroring involves duplication of data — the amount of storage capacity needed is twice the amount of data being stored.
- Therefore, mirroring is considered expensive and is preferred for mission-critical applications that cannot afford data loss.
- Mirroring improves read performance because read requests can be serviced by both disks.
- However, write performance deteriorates, as each write request manifests as two writes on the HDDs.
- In other words, mirroring does not deliver the same levels of write performance as a striped RAID.

3.3.3 Parity:

- Parity is a method of protecting striped data from HDD failure without the cost of mirroring.
- An additional HDD is added to the stripe width to hold parity, a mathematical construct that allows re-creation of the missing data.
- Parity is a redundancy check that ensures full protection of data without maintaining a full set of duplicate data.
- Parity information can be stored on separate, dedicated HDDs or distributed across all the drives in a RAID set.
- Figure 3-4 shows a parity RAID.
- The first four disks, labeled D, contain the data.
- The fifth disk, labeled P, stores the parity information, which in this case is the sum of the elements in each row.
- Now, if one of the Ds fails, the missing value can be calculated by subtracting the sum of the rest of the elements from the parity value.

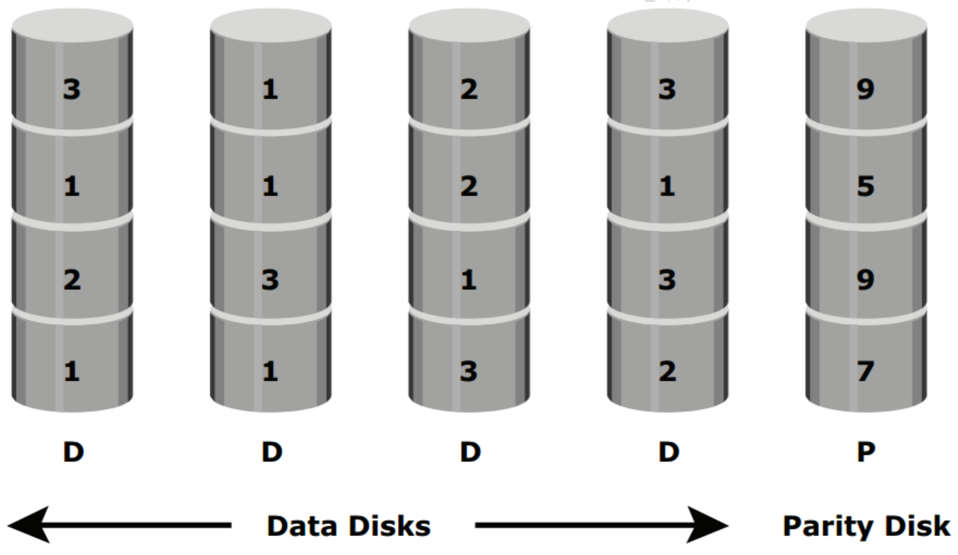


Figure 3-4: Parity RAID

- In Figure 3-4, the computation of parity is represented as a simple arithmetic operation on the data.
- However, parity calculation is a bitwise XOR operation.
- Calculation of parity is a function of the RAID controller.
- Compared to mirroring, parity implementation considerably reduces the cost associated with data protection.
- Consider a RAID configuration with five disks.
- Four of these disks hold data, and the fifth holds parity information.
- Parity requires 25 percent extra disk space compared to mirroring, which requires 100 percent extra disk space.
- However, there are some disadvantages of using parity.

- Parity information is generated from data on the data disk.
- Therefore, parity is recalculated every time there is a change in data.
- This recalculation is time consuming and affects the performance of the RAID controller.

RAID Levels

- RAID levels (see Table 3-1) are defined on the basis of striping, mirroring, and parity techniques.
- These techniques determine the data availability and performance characteristics of an array.
- Some RAID arrays use one technique, whereas others use a combination of techniques.
- Application performance and data availability requirements determine the RAID level selection.

Table 3-1: Raid Levels

LEVELS	BRIEF DESCRIPTION
RAID 0	Striped array with no fault tolerance
RAID 1	Disk mirroring
RAID 3	Parallel access array with dedicated parity disk
RAID 4	Striped array with independent disks and a dedicated parity disk
RAID 5	Striped array with independent disks and distributed parity
RAID 6	Striped array with independent disks and dual distributed parity
Nested	Combinations of RAID levels. Example: RAID 1 + RAID 0

RAID 0

- In a RAID 0 configuration, data is striped across the HDDs in a RAID set.
- It utilizes the full storage capacity by distributing strips of data over multiple HDDs in a RAID set.
- To read data, all the strips are put back together by the controller.
- The stripe size is specified at a host level for software RAID and is vendor specific for hardware RAID.
- Figure 3-5 shows RAID 0 on a storage array in which data is striped across 5 disks.
- When the number of drives in the array increases, performance improves because more data can be

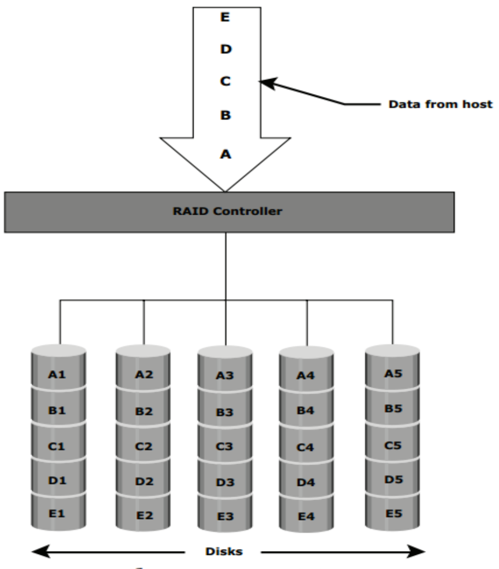


Figure 3-5: RAID 0

read or written simultaneously.

- RAID 0 is used in applications that need high I/O throughput.
- However, if these applications require high availability, RAID 0 does not provide data protection and availability in the event of drive failures.

RAID 1

- In a RAID 1 configuration, data is mirrored to improve fault tolerance (see Figure 3-6).
- A RAID 1 group consists of at least two HDDs.
- As explained in mirroring, every write is written to both disks, which is transparent to the host in a hardware RAID implementation.
- In the event of disk failure, the impact on data recovery is the least among all RAID implementations.
- This is because the RAID controller uses the mirror drive for data recovery and continuous operation.
- RAID 1 is suitable for applications that require high availability.

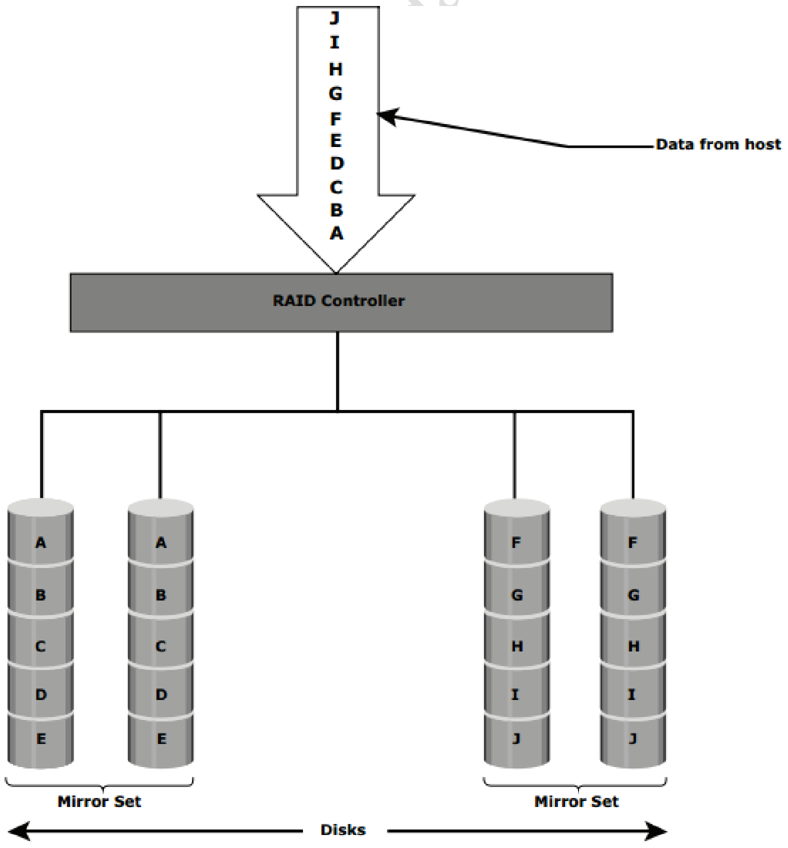


Figure 3-6: RAID 1

Nested RAID

- Most data centers require data redundancy and performance from their RAID arrays.
- RAID 0+1 and RAID 1+0 combine the performance benefits of RAID 0 with the redundancy benefits of RAID 1.
- They use striping and mirroring techniques and combine their benefits.
- These types of RAID require an even number of disks, the minimum being four (see Figure 3-7).
- RAID 1+0 is also known as RAID 10 (Ten) or RAID 1/0.
- Similarly, RAID 0+1 is also known as RAID 01 or RAID 0/1.
- RAID 1+0 performs well for workloads that use small, random, write-intensive I/O.

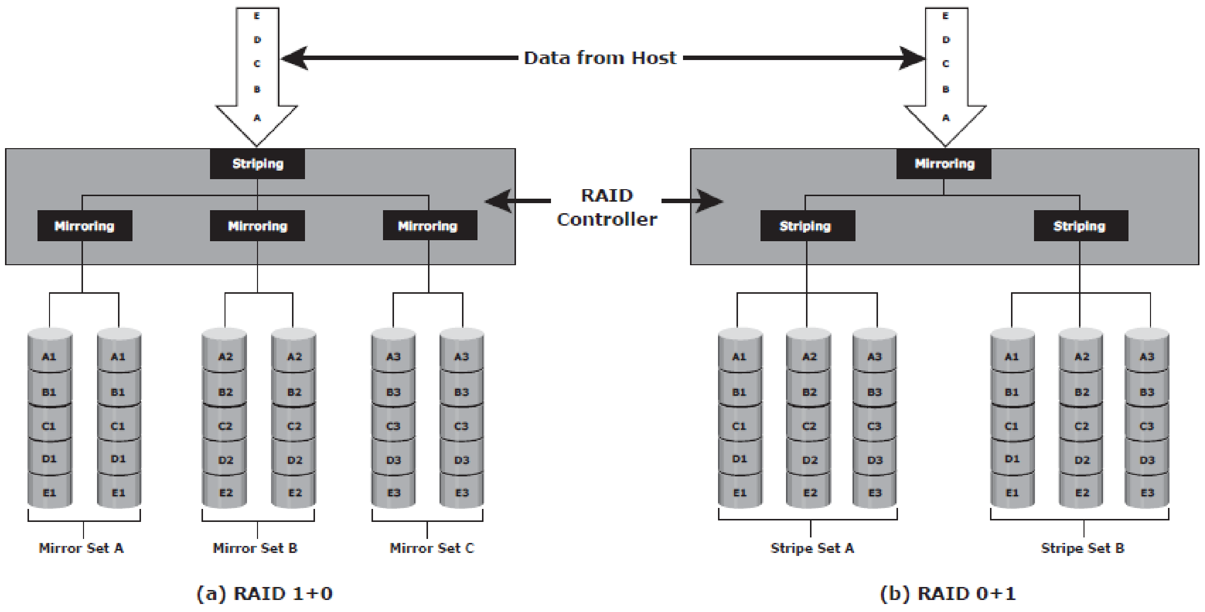


Figure 3-7: Nested RAID

- Some applications that benefit from RAID 1+0 include the following:
  - ➢ High transaction rate Online Transaction Processing (OLTP)
  - ➢ Large messaging installations
  - ➢ Database applications that require high I/O rate, random access, and high availability
- A common misconception is that RAID 1+0 and RAID 0+1 are the same.
- Under normal conditions, RAID levels 1+0 and 0+1 offer identical benefits.
- However, rebuild operations in the case of disk failure differ between the two.
- RAID 1+0 is also called **striped mirror**.
- The basic element of RAID 1+0 is a mirrored pair, which means that data is first mirrored and then both copies of data are striped across multiple HDDs in a RAID set.

- When replacing a failed drive, only the mirror is rebuilt.
- In other words, the disk array controller uses the surviving drive in the mirrored pair for data recovery and continuous operation.
- Data from the surviving disk is copied to the replacement disk.
- RAID 0+1 is also called mirrored stripe.
- The basic element of RAID 0+1 is a stripe.
- This means that the process of striping data across HDDs is performed initially and then the entire stripe is mirrored.
- If one drive fails, then the entire stripe is faulted.
- A rebuild operation copies the entire stripe, copying data from each disk in the healthy stripe to an equivalent disk in the failed stripe.
- This causes increased and unnecessary I/O load on the surviving disks and makes the RAID set more vulnerable to a second disk failure

RAID 3

- RAID 3 stripes data for high performance and uses parity for improved fault tolerance.
- Parity information is stored on a dedicated drive so that data can be reconstructed if a drive fails.
- For example, of five disks, four are used for data and one is used for parity.
- Therefore, the total disk space required is 1.25 times the size of the data disks.
- RAID 3 always reads and writes complete stripes of data across all disks, as the drives operate in parallel.
- There are no partial writes that update one out of many strips in a stripe.
- Figure 3-8 illustrates the RAID 3 implementation.

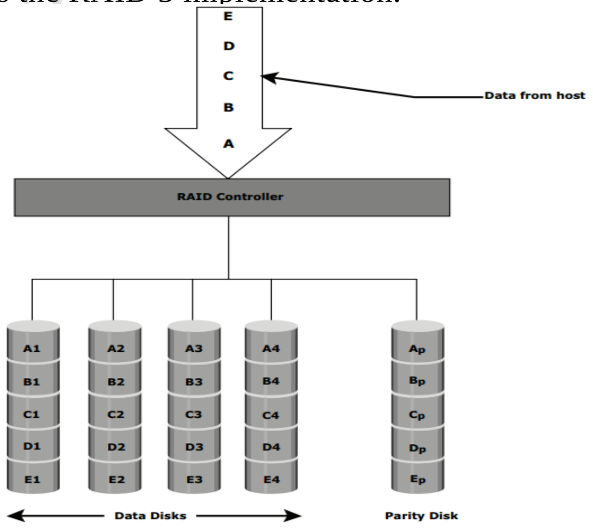


Figure 3-8: RAID 3

- RAID 3 provides good bandwidth for the transfer of large volumes of data.
- RAID 3 is used in applications that involve large sequential data access, such as video streaming.

RAID 4

- Similar to RAID 3, RAID 4 stripes data for high performance and uses parity for improved fault tolerance (refer to Figure 3-8).
- Data is striped across all disks except the parity disk in the array.
- Parity information is stored on a dedicated disk so that the data can be rebuilt if a drive fails.
- Striping is done at the block level.
- Unlike RAID 3, data disks in RAID 4 can be accessed independently so that specific data elements can be read or written on single disk without read or write of an entire stripe.
- RAID 4 provides good read throughput and reasonable write throughput.

RAID 5

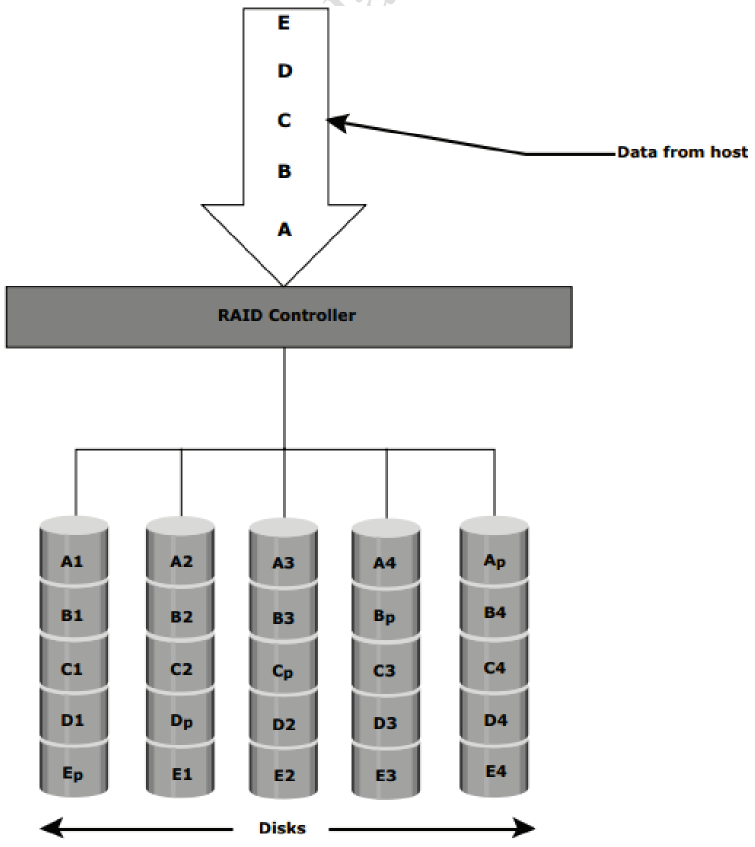


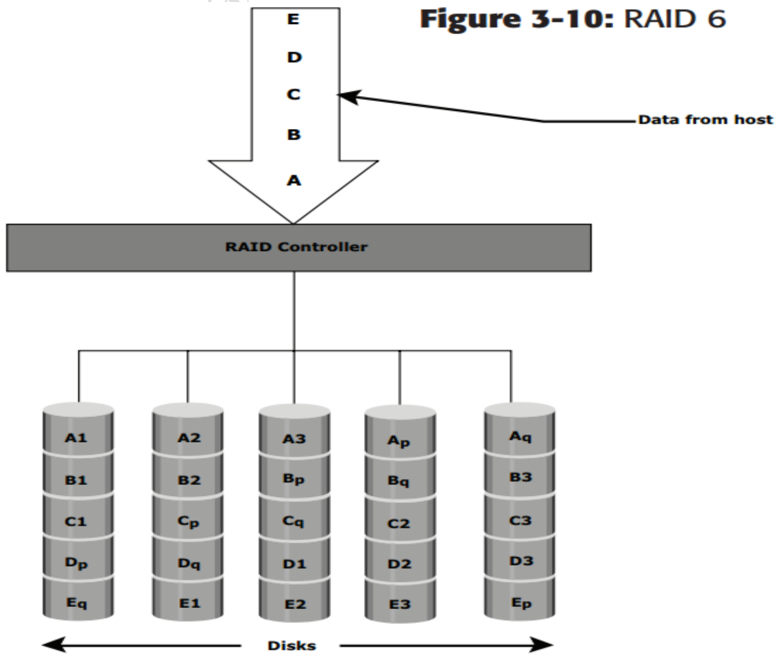
Figure 3-9: RAID 5

- RAID 5 is a very versatile RAID implementation.
- It is similar to RAID 4 because it uses striping and the drives (strips) are independently accessible.
- The difference between RAID 4 and RAID 5 is the parity location.
- In RAID 4, parity is written to a dedicated drive, creating a write bottleneck for the parity disk.
- In RAID 5, parity is distributed across all disks.
- The distribution of parity in RAID 5 overcomes the write bottleneck.
- Figure 3-9 illustrates the RAID 5 implementation.
- RAID 5 is preferred for messaging, data mining, medium-performance media serving, and relational database management system (RDBMS) implementations in which database administrators (DBAs) optimize data access.

RAID 6

- RAID 6 works the same way as RAID 5 except that RAID 6 includes a second parity element to enable survival in the event of the failure of two disks in a RAID group (Figure 3-10).
- Therefore, a RAID 6 implementation requires at least four disks.
- RAID 6 distributes the parity across all the disks.
- The write penalty in RAID 6 is more than that in RAID 5; therefore, RAID 5 writes perform better than RAID 6.
- The rebuild operation in RAID 6 may take longer than that in RAID 5 due to the presence of two parity sets.

Figure 3-10: RAID 6



RAID Impact on Disk Performance:

- When choosing a RAID type, it is imperative to consider the impact to disk performance and application IOPS.
- In both mirrored and parity RAID configurations, every write operation translates into more I/O overhead for the disks which is referred to as write penalty.
- In a RAID 1 implementation, every write operation must be performed on two disks configured as a mirrored pair while in a RAID 5 implementation, a write operation may manifest as four I/O operations.
- When performing small I/Os to a disk configured with RAID 5, the controller has to read, calculate, and write a parity segment for every data write operation.
- Figure 3-11 illustrates a single write operation on RAID 5 that contains a group of five disks.
- Four of these disks are used for data and one is used for parity.
- The parity (P) at the controller is calculated as follows:

EP=E1+E2+E3+E4 (XOR operations)

- Here, D1 to D4 is striped data across the RAID group of five disks.
- Whenever the controller performs a write I/O, parity must be computed by reading the old parity (Ep old) and the old data (E4 old) from the disk, which means two read I/Os.
- The new parity (Ep new) is computed as follows:

$$E_{p\ new} = E_{p\ old} - E_{4\ old} + E_{4\ new}\ \textbf{(XOR operations)}$$

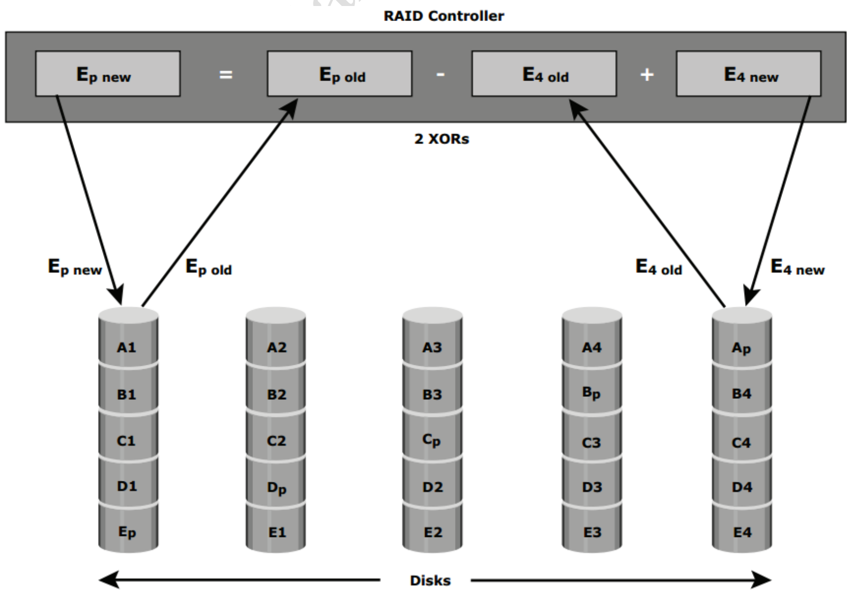


Figure 3-11: Write penalty in RAID 5

- After computing the new parity, the controller completes the write I/O by writing the new data and the new parity onto the disks, amounting to two write I/Os.

- Therefore, the controller performs two disk reads and two disk writes for every write operation, and the write penalty in RAID 5 implementations is 4.
- In RAID 6, which maintains dual parity, a disk write requires three read operations: for Ep1 old, Ep2 old, and E4 old.
- After calculating Ep1 new and Ep2 new, the controller performs three write I/O operations for Ep1 new, Ep2 new and E4 new.
- Therefore, in a RAID 6 implementation, the controller performs six I/O operations for each write I/O, and the write penalty is 6.

3.5.1 Application IOPS and RAID Configurations

- When deciding the number of disks required for an application, it is important to consider the impact of RAID based on IOPS generated by the application.
- The total disk load should be computed by considering the type of RAID configuration and the ratio of read compared to write from the host.
- The following example illustrates the method of computing the disk load in different types of RAID.
- Consider an application that generates 5,200 IOPS, with 60 percent of them being reads.
- The disk load in RAID 5 is calculated as follows:
  RAID 5 disk load = 0.6 × 5,200 + 4 × (0.4 × 5,200)
  [because the write penalty for RAID 5 is 4]
  = 3,120 + 4 × 2,080
  = 3,120 + 8,320
  = 11,440 IOPS

The disk load in RAID 1 is calculated as follows:

- RAID 1 disk load = 0.6 × 5,200 + 2 × (0.4 × 5,200)
  [because every write manifests as two writes to the disks]
  = 3,120 + 2 × 2,080
  = 3,120 + 4,160
  = 7,280 IOPS
- The computed disk load determines the number of disks required for the application.
- If in this example an HDD with a specification of a maximum 180 IOPS for the application needs to be used, the number of disks required to meet the workload for the RAID configuration would be as follows:

  - **RAID 5:** 11,440 / 180 = 64 disks
  - **RAID 1:** 7,280 / 180 = 42 disks (approximated to the nearest even number)

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 the 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 cache.

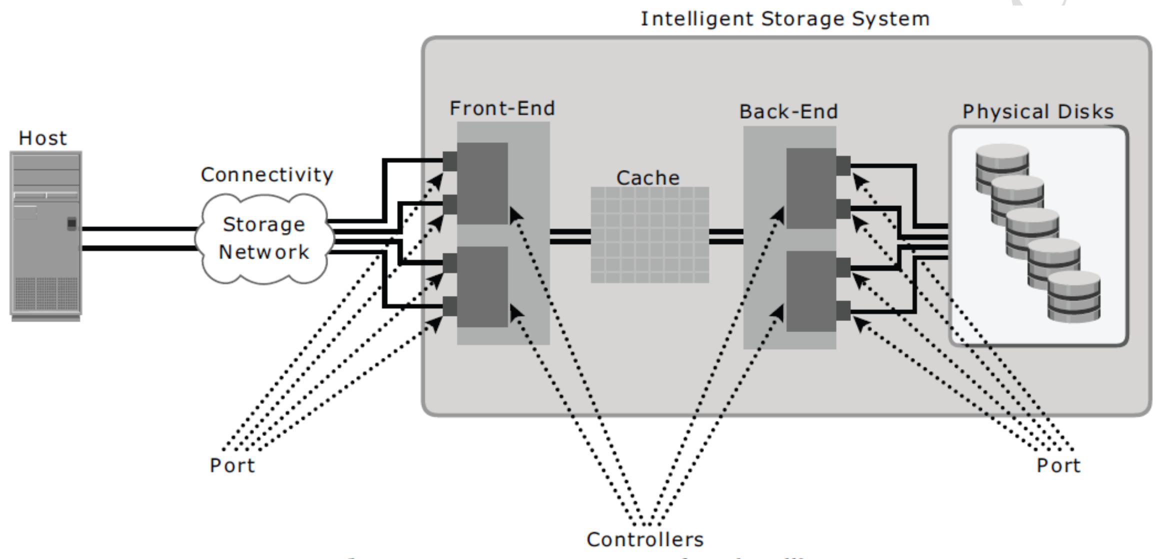


Figure 4-1: Components of an intelligent storage system

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.
- The front-end ports enable hosts to connect to the intelligent storage system.
- Each front-end port has processing logic that executes the appropriate transport protocol, such as SCSI, Fibre Channel, or iSCSI, for storage connections.
- Redundant ports are provided on the front end for high availability.
- Front-end controllers route data to and from cache via the internal data bus.
- When cache receives write data, the controller sends an acknowledgment message back to the host.
- Controllers optimize I/O processing by using command queuing algorithms.

Front-End Command Queuing

- Command queuing is a technique implemented on front-end controllers.
- It determines the execution order of received commands and can reduce unnecessary drive head movements and improve disk performance.
- When a command is received for execution, the command queuing algorithms assigns a tag that defines a sequence in which commands should be executed.

- With command queuing, multiple commands can be executed concurrently based on the organization of data on the disk, regardless of the order in which the commands were received.
- The most commonly used command queuing algorithms are as follows:

➢ **First In First Out (FIFO):** This is the default algorithm where commands are executed in the order in which they are received (Figure 4-2 [a]). There is no reordering of requests for optimization; therefore, it is inefficient in terms of performance.

➢ **Seek Time Optimization:** Commands are executed based on optimizing read/write head movements, which may result in reordering of commands. Without seek time optimization, the commands are executed in the order they are received. For example, as shown in Figure 4-2(a), the commands are executed in the order A, B, C and D. The radial movement required by the head to execute C immediately after A is less than what would be required to execute B. With seek time optimization, the command execution sequence would be A, C, B and D, as shown in Figure 4-2(b).

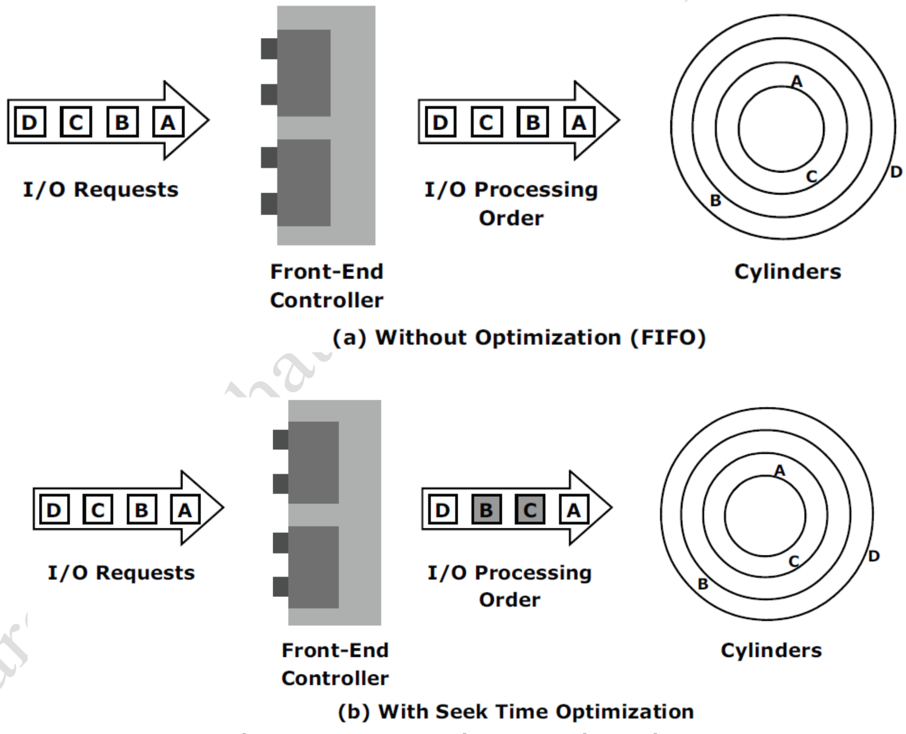


Figure 4-2: Front-end command queuing

➢ **Access Time Optimization:** Commands are executed based on the combination of seek time optimization and an analysis of rotational latency for optimal performance.

- Command queuing can also be implemented on disk controllers and this may further supplement the command queuing implemented on the front-end controllers.
- Some models of SCSI and Fibre Channel drives have command queuing implemented on their controllers.

4.1.2 Cache

- Cache is an important component that enhances the I/O performance in an intelligent storage system.
- Cache is semiconductor memory where data is placed temporarily to reduce the time required to service I/O requests from the host.
- Cache **improves storage system performance** by isolating hosts from the mechanical delays associated with physical disks, which are the slowest components of an intelligent storage system.
- Accessing data from a physical disk usually takes a few milliseconds because of seek times and rotational latency.
- If a disk has to be accessed by the host for every I/O operation, requests are queued, which results in a delayed response.
- Accessing data from cache takes less than a millisecond.
- Write data is placed in cache and then written to disk.
- After the data is securely placed in cache, the host is acknowledged immediately.

Structure of Cache

- Cache is organized into pages or slots, 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 while tag RAM tracks the location of the data in the data store (see Figure 4-3) and in disk.
- 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 or not.
- 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.

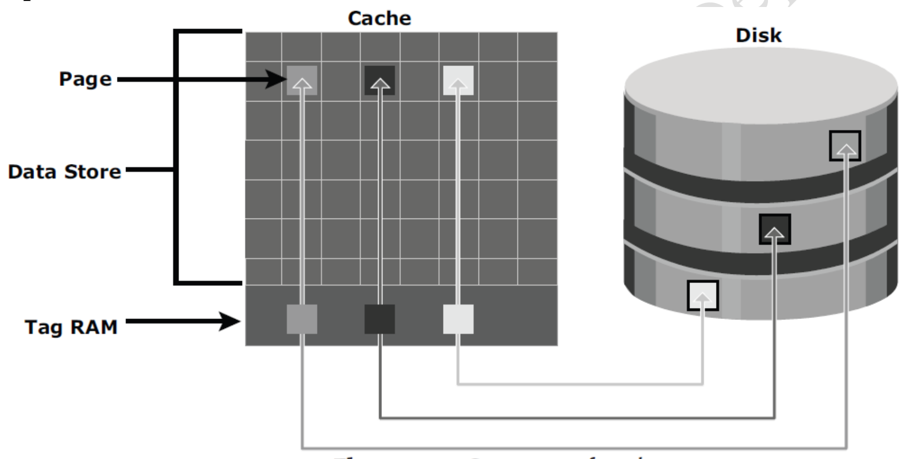
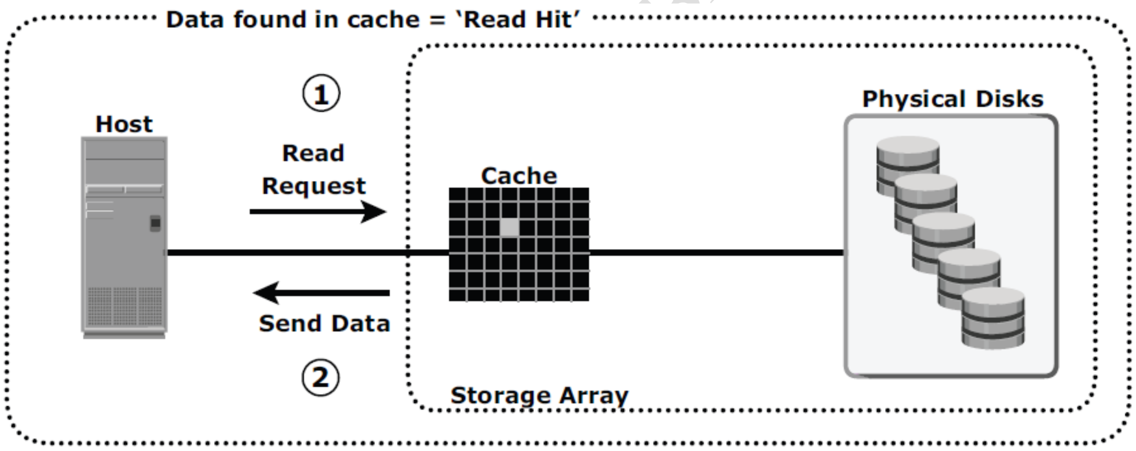


Figure 4-3: Structure of cache

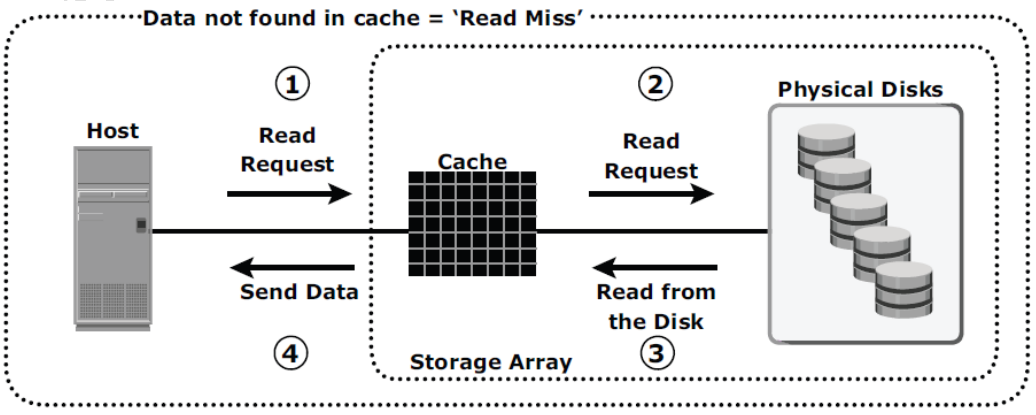
Read Operation with Cache

- When a host issues a read request, the front-end controller accesses 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 4-4[a]).
- This provides 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 4-4[b]).
- The back-end controller accesses the appropriate disk and retrieves the requested data.
- Data is then placed in cache and is finally sent to the host through the front-end controller.
- Cache misses increase I/O response time.
- A pre-fetch, 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 pre-fetch, the intelligent storage system pre-fetches a fixed amount of data.
- It is most suitable when I/O sizes are uniform.
- In variable pre-fetch, 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 pre-fetched to prevent the disks from being rendered busy with pre-fetch at the expense of other I/O.
- 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.



(a) Read Hit



(b) Read Miss

Figure 4-4: Read hit and read miss

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, as the write operations are isolated from the mechanical delays of the disk. However, uncommitted data is at risk of loss in the event of cache failures.
- **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 write response time is longer because of the disk operations.
- Cache can be bypassed under certain conditions, such as very 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 area.
- This is particularly useful in an environment where cache resources are constrained and must be made available 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, as only one global set of addresses has to be managed.
- Global cache may allow users to specify the percentages of cache available for reads and writes in cache management.
- Typically, the read cache is small, but 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 intelligent storage systems can be configured with large amounts 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:

➢ **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 which hasn't 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, data will first be written to disk before the page is reused.

➢ **Most Recently Used (MRU):** An algorithm that is the converse of LRU. In MRU, 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) in order to manage its availability.
- Flushing is the process of committing 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 the high-speed or forced flushing and returns to idle flush behavior.
- The cache utilization level, as shown in Figure 4-5, 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 to flushing. This type of flushing has minimal impact on host 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, dirty pages are forcibly flushed to disk.

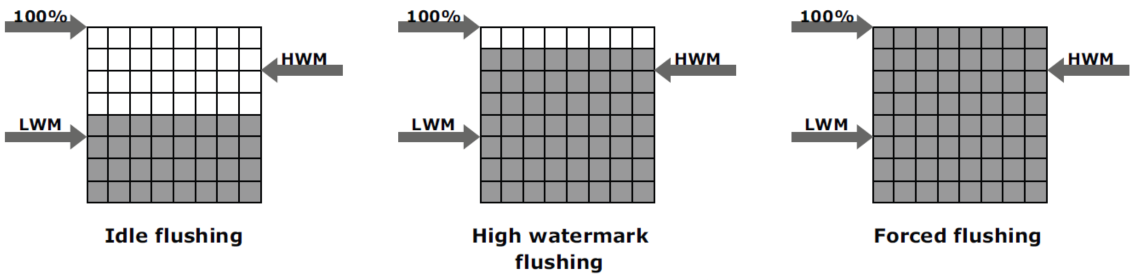


Figure 4-5: Types of flushing

Cache Data Protection

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

<u>Cache mirroring:</u>

- Each write to cache is held in two different memory locations on two independent memory cards.
- In the event of a cache failure, 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, in the event of a cache failure, the data can still be accessed from the disk.
- As 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.

<u>Cache vaulting:</u>

- Cache is exposed to the risk of uncommitted data loss due to power failure.
- This problem can be addressed in various ways: powering the memory with a battery until AC power is restored or using battery power to write the cache content to the disk.
- In the event of extended power failure, using batteries is not a viable option because in intelligent storage systems, large amounts of data may need to be committed to numerous disks and batteries may 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.

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 availability, storage systems are configured with dual controllers with multiple ports.
- Such configurations provide an alternate path to physical disks in the event of a controller or port failure.
- 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.

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.

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.

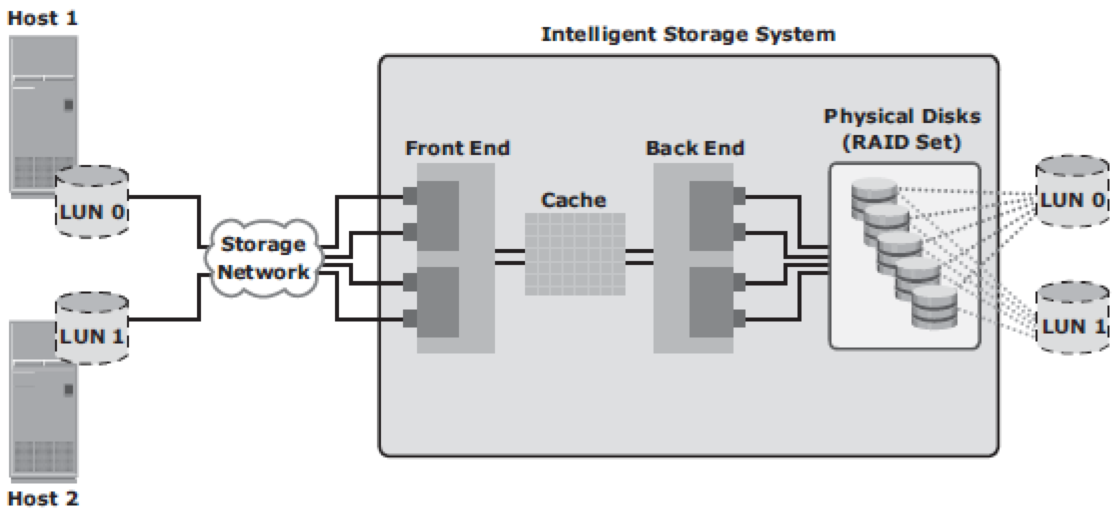


Figure 4-5: RAID set and LUNs

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 fi le 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 fi le system, and then virtual disks are created on it. Virtual disks are files on the hypervisor 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.

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.

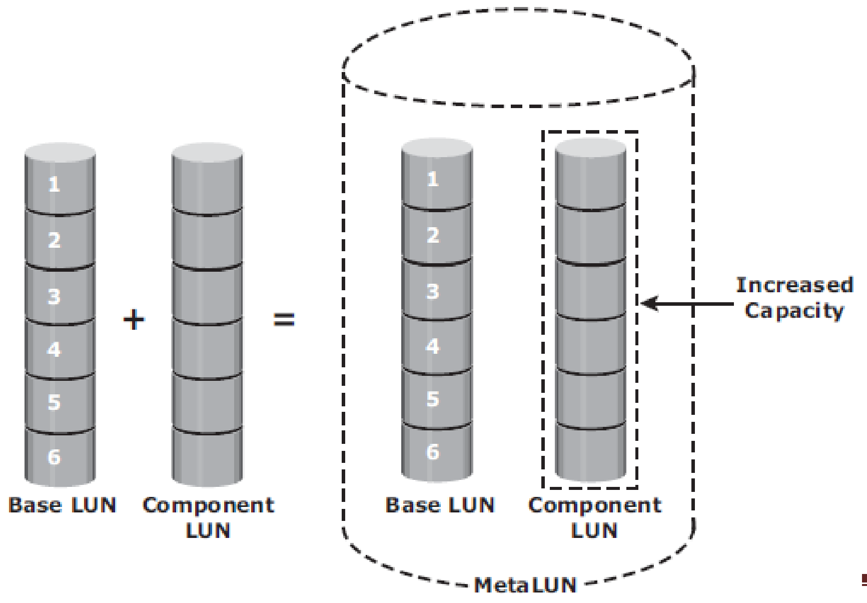


Figure 4-6: Concatenated metaLUN

Reference: Information Storage and Management,Author :EMC Education Services, Publisher: Wiley ISBN: 9781118094839

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.)

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).

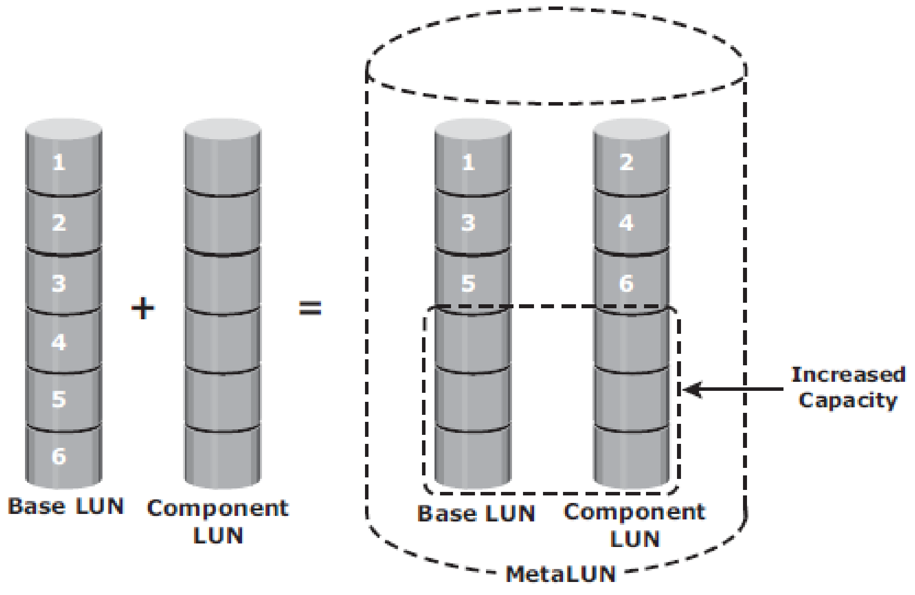


Figure 4-7: Striped metaLUN

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

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.

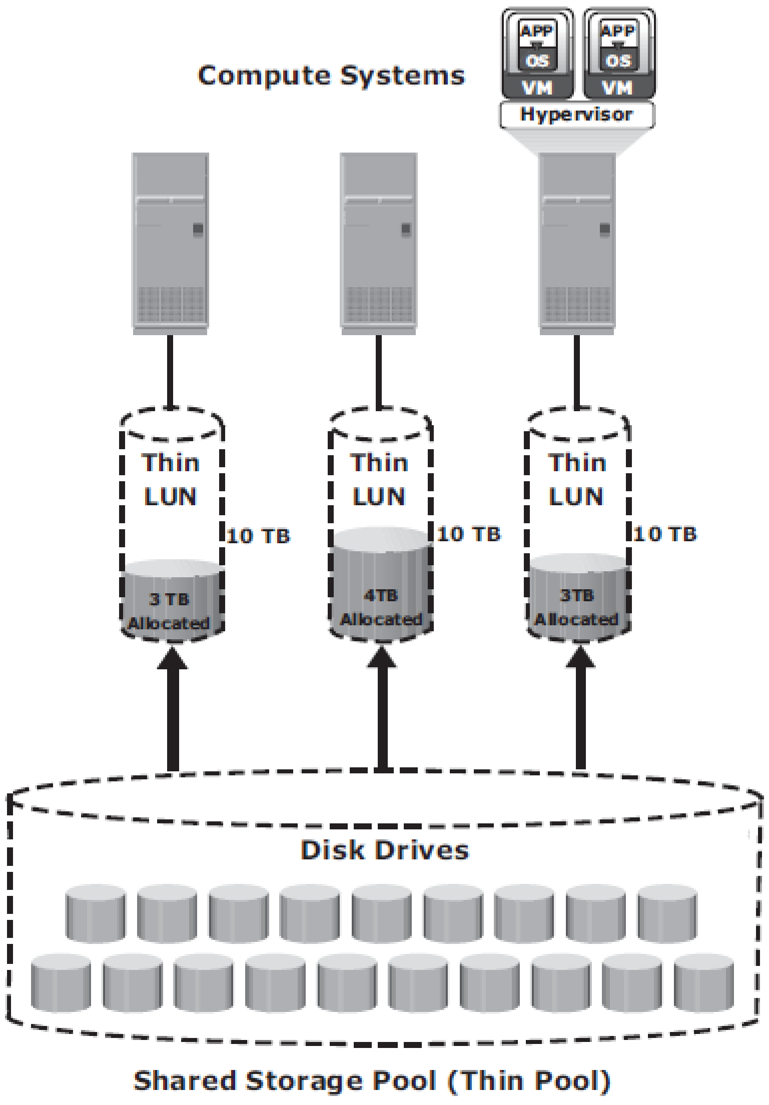


Figure 4-8: Virtual provisioning

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.

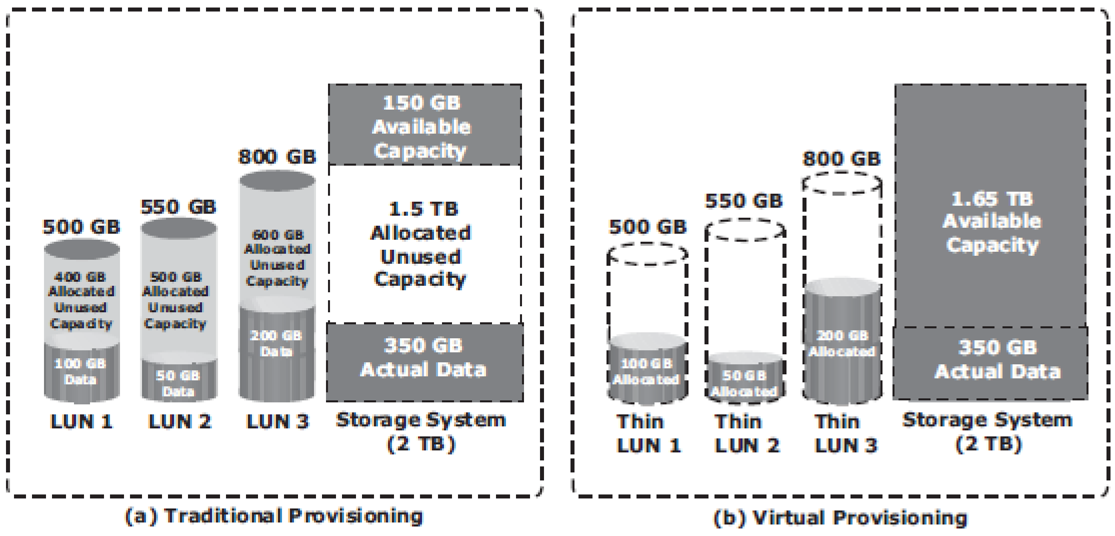


Figure 4-9: Traditional versus virtual provisioning

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.

LUN Masking

- LUN masking is a process that provides data access control by defining which LUNs a host can access.
- LUN masking function is typically implemented at the front end controller.
- This ensures that volume access by servers is controlled appropriately, preventing unauthorized or accidental use in a distributed environment.
- For example, consider a storage array with two LUNs that store data of the sales and finance departments.
- Without LUN masking, both departments 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.

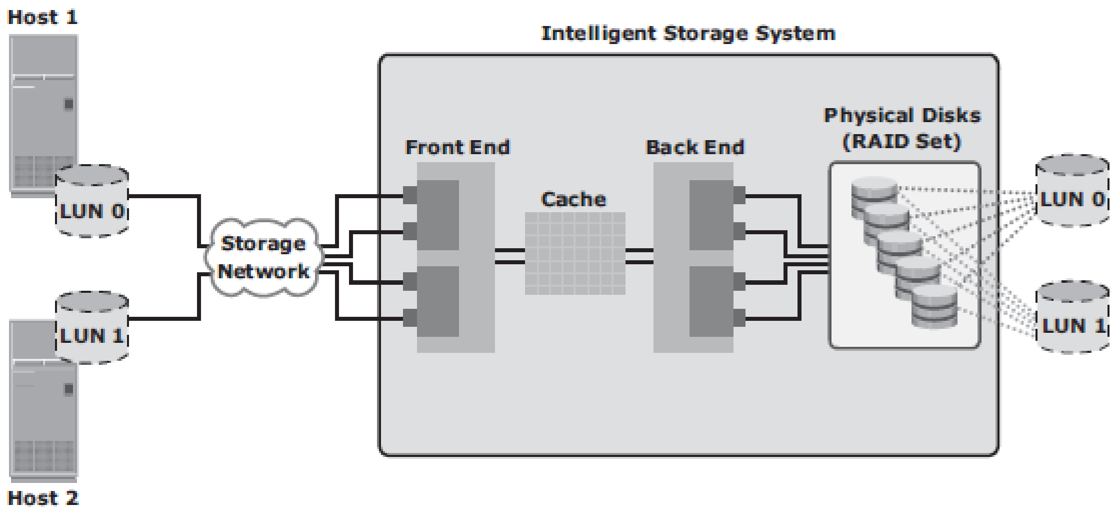


Figure 4-5: RAID set and LUNs