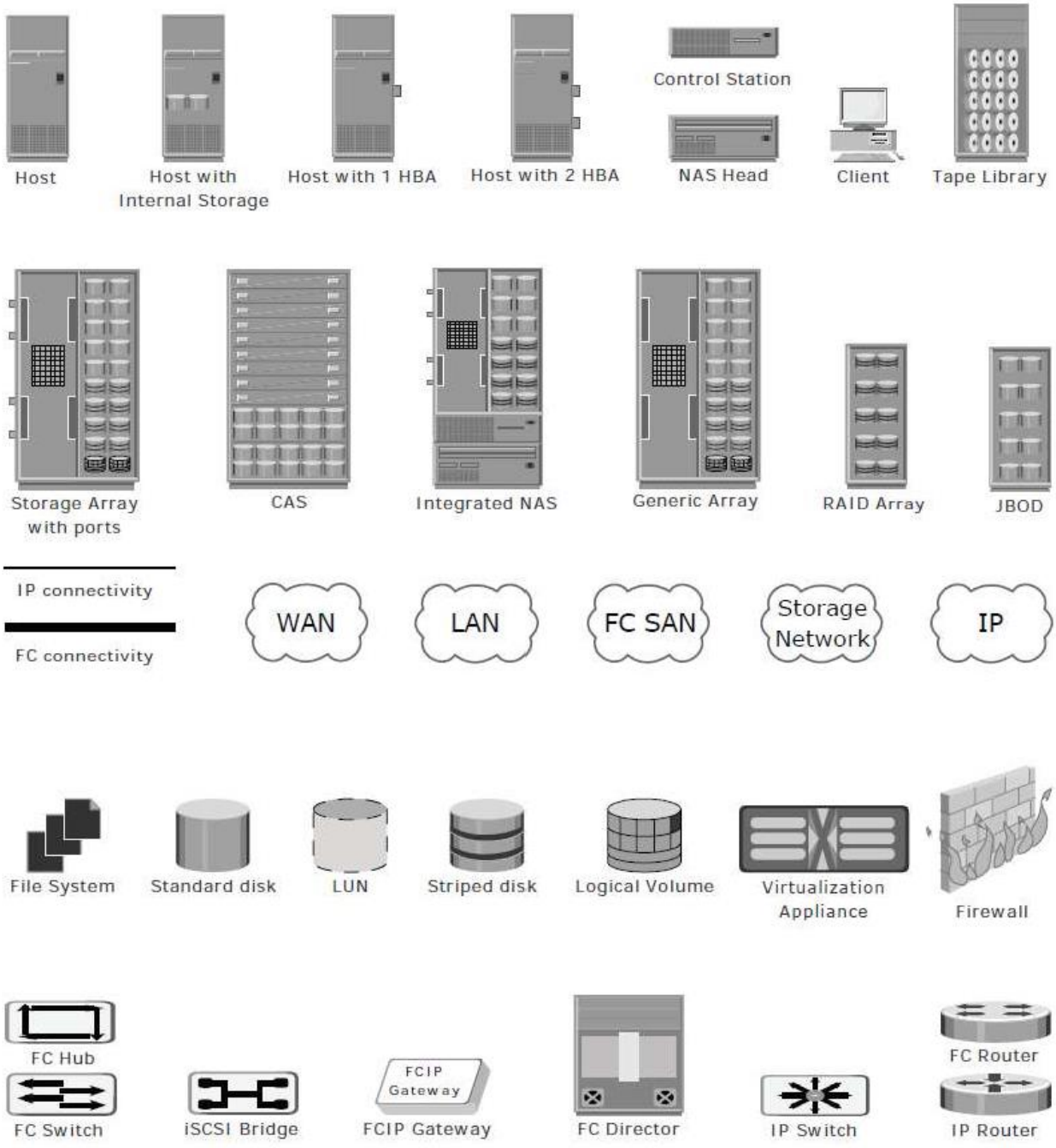
Module 1

STORAGE SYSTEM

1.1 Introduction to Information storage

Icons Used In The Notes



1.1.1 Why Information management?

- Information is increasingly important in our daily lives. We have become information Dependents.
- We live in on-command, on-demand world that means we need information when and where it is required.
- We access the Internet every day to perform searches, participate in social networking, send and receive e-mails, share pictures and videos, and scores of other applications. Equipped with a growing number of content-generating devices, more information is being created by individuals than by businesses.
- The importance, dependency, and volume of information for the business world also continue to grow at astounding rates.
- Businesses depend on fast and reliable access to information critical to their success. Some of the business applications that process information include airline reservations, telephone billing systems, e-commerce, ATMs, product designs, inventory management, e-mail archives, Web portals, patient records, credit cards, life sciences, and global capital markets.
- The increasing criticality of information to the businesses has amplified the challenges in protecting and managing the data.
- Organizations maintain one or more data centers to store and manage information. A data center is a facility that contains information storage and other physical information technology (IT) resources for computing, networking, and storing information.

1.1.2 Information Storage

Businesses use data to derive information that is critical to their day-to-day operations. Storage is a repository that enables users to store and retrieve this digital data.

Data

- Data is a collection of raw facts from which conclusions may be drawn.

- Eg: a printed book, a family photograph, a movie on videotape, e-mail message, an e-book, a bitmapped image, or a digital movie are all examples of data.

- The data can be generated using a computer and stored in strings of 0s and 1s(as shown in Fig 1.1), is called digital data and is accessible by the user only after it is processed by a computer.

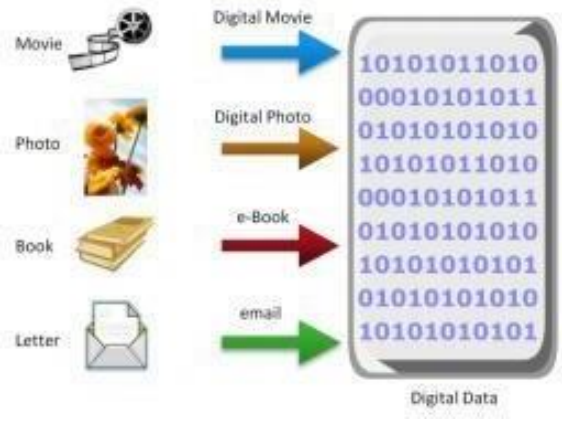


Fig 1.1: Digital data

The following is a list of some of the factors that have contributed to the growth of digital data :

1. **Increase in data processing capabilities**: Modern-day computers provide a significant increase in processing and storage capabilities. This enables the conversion of various types of content and media from conventional forms to digital formats.
2. **Lower cost of digital storage:** Technological advances and decrease in the cost of storage devices have provided low-cost solutions and encouraged the development of less expensive data storage devices. This cost benefit has increased the rate at which data is being generated and stored.
3. **Affordable and faster communication technology:** The rate of sharing digital data is now much faster than traditional approaches. A handwritten letter may take a week to reach its destination, whereas it only takes a few seconds for an e-mail message to reach its recipient.

4. **Proliferation of applications and smart devices:** Smartphones, tablets, and newer digitaldevices, along with smart applications, have significantly contributed to the generation of digital content.

1.1.3 Types of Data

Data can be classified as structured or unstructured (see Fig 1.2) based on how it is stored and managed.

- **Structured data:**
  - Structured data is organized in rows and columns in a rigidly defined format so that applications can retrieve and process it efficiently.
  - Structured data is typically stored using a database management system (DBMS).
- **Unstructured data:**
  - Data is unstructured if its elements cannot be stored in rows and columns, and is therefore difficult to query and retrieve by business applications.
  - Example: e-mail messages, business cards, or even digital format files such as .doc, .txt, and .pdf.

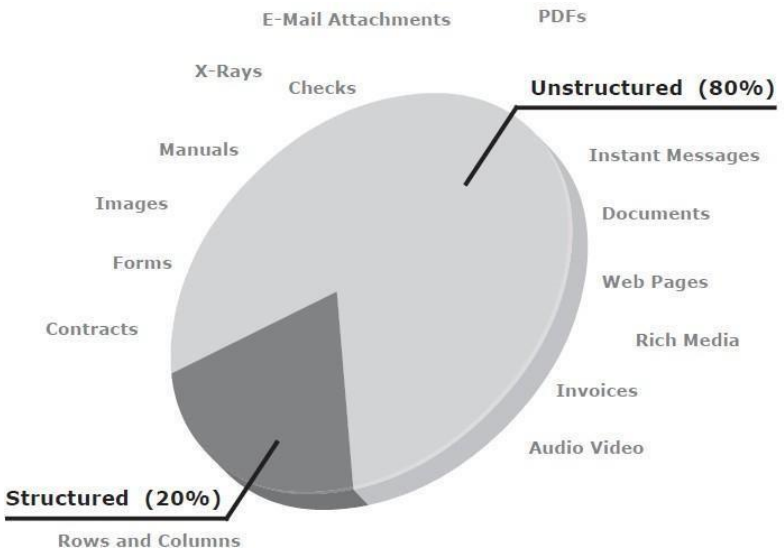


Fig 1.2: Types of data

1.1.4 Big Data

- Big data refers to data sets whose sizes are beyond the capability of commonly used software tools to capture, store, manage, and process within acceptable time limits.

- It includes both structured and unstructured data generated by a variety of sources, including business application transactions, web pages, videos, images, e-mails, social media, and so on.
- The big data ecosystem (see Fig 1.3) consists of the following:
  1. Devices that collect data from multiple locations and also generate new data about this data (metadata).
  2. Data collectors who gather data from devices and users.
  3. Data aggregators that compile the collected data to extract meaningful information.
  4. Data users and buyers who benefit from the information collected and aggregated by others in the data value chain .

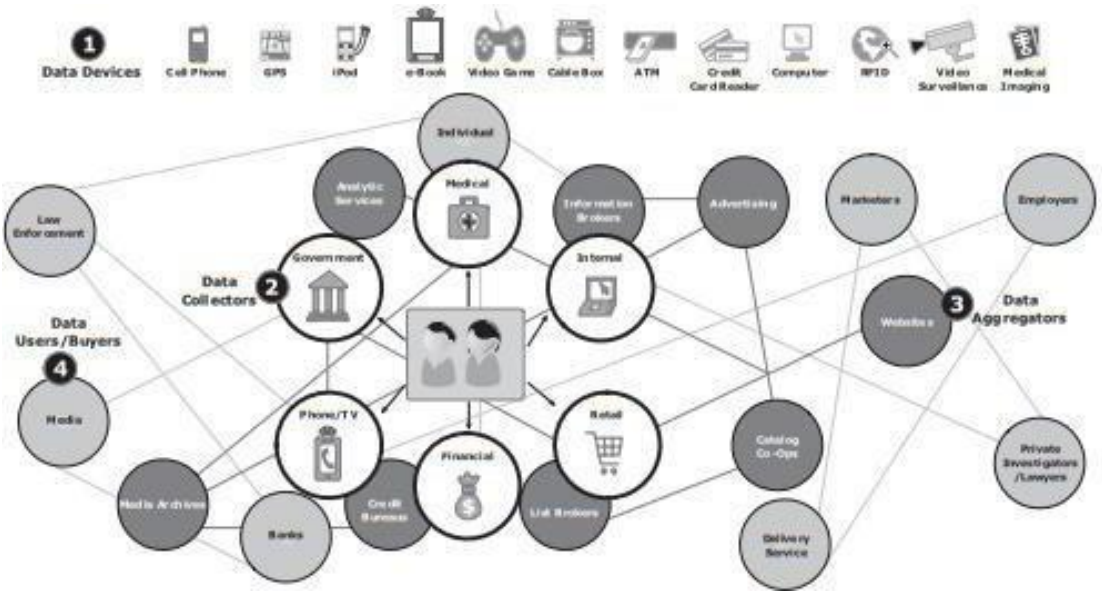


Fig 1.3: Big data Ecosystem

- Big data Analysis in real time requires new techniques, architectures, and tools that provide :
  1. high performance,
  2. massively parallel processing (MPP) data platforms,
  3. advanced analytics on the data sets.
- Big data Analytics provide an opportunity to translate large volumes of data into right decisions.

1.1.5 Information

- Data, whether structured or unstructured, does not fulfil any purpose for individuals or businesses unless it is presented in a meaningful form.

- Information is the intelligence and knowledge derived from data.
- Businesses analyze raw data in order to identify meaningful trends. On the basis of these trends, a company can plan or modify its strategy.
- For example, a retailer identifies customers' preferred products and brand names by analyzing their purchase patterns and maintaining an inventory of those products.
- Because information is critical to the success of a business, there is an ever present concern about its availability and protection.

1.1.6 Storage

- Data created by individuals or businesses must be stored so that it is easily accessible for further processing.
- In a computing environment, devices designed for storing data are termed storage devices or simply storage.
- The type of storage used varies based on the type of data and the rate at which it is created and used.
  - Devices such as memory in a cell phone or digital camera, DVDs, CD-ROMs, and hard disks in personal computers are examples of storage devices.
- Businesses have several options available for storing data including internal hard disks, external disk arrays and tapes.

1.2 Introduction to Evolution 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. This approach is referred to as **server-centric storage architecture** (see Fig 1.4 [a]).
- In this server-centric storage 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 rapid increase in the number of departmental servers in an enterprise resulted in

unprotected, unmanaged, fragmented islands of information and increased capital and operating expenses.

- To overcome these challenges, storage evolved from **server-centric to information-centric architecture** (see Fig 1.4 [b]).

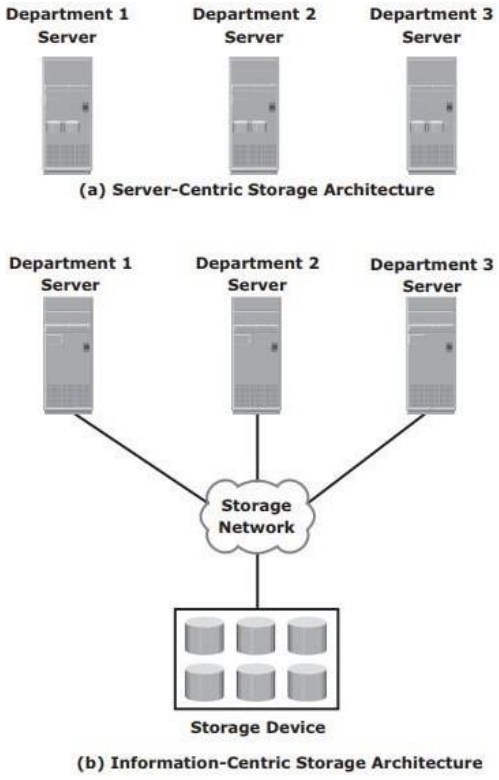


Fig 1.4: Evolution of storage architecture

- In information-centric 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 continues to evolve, which enables organizations to

consolidate, protect, optimize, and leverage their data to achieve the highest return on information assets.

1.3 Data Center Infrastructure

- Organizations maintain data centers to provide centralized data processing capabilities across the enterprise.
- The data center infrastructure includes computers, storage systems, network devices, dedicated power backups, and environmental controls (such as air conditioning and fire suppression).

1.3.1 Key Data Center Elements

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

1) **Application:** An application is a computer program that provides the logic for computing operations. Eg: order processing system.
2) **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.
3) **Host or compute:** A computing platform (hardware, firmware, and software) that runs applications and databases.
4) **Network:** A data path that facilitates communication among various networked devices.
5) **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.
- Fig 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 a client machine connected over a LAN/ WAN to a host running an order-processing application.
    2) The client accesses the DBMS on the host through the application to provide order-related information, such as the customer name, address, payment method, products ordered, and quantity ordered.

3) The DBMS uses the host operating system to write this data to the database located on physical disks in the storage array.
4) The Storage Network provides the communication link between the host and the storage array and transports the request to read or write commands between them.
5) The storage array, after receiving the read or write request from the host, performs the necessary operations to store the data on physical disks.

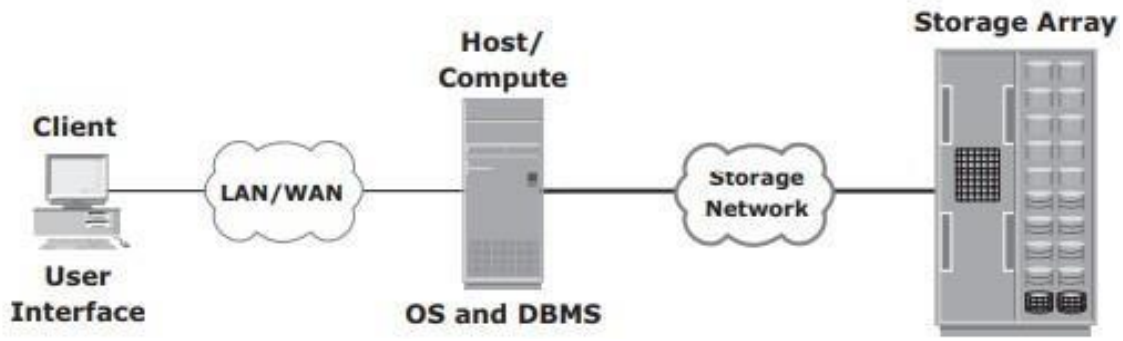


Fig 1.5: Example of an online order transaction system

1.4 Key characteristics for Data Center Elements

Key characteristics of data center elements are:

1) **Availability:** All data center elements should be designed to ensure accessibility. The inability of users to access data can have a significant negative impact on a business.
2) **Security:** Polices, procedures, and proper integration of the data center core elements that will prevent unauthorized access to information must be established. Specific mechanisms must enable servers to access only their allocated resources on storage arrays.
3) **Scalability:** Data center operations should be able to allocate additional processing capabilities (eg: servers, new applications, and additional databases) or storage on demand, without interrupting business operations. The storage solution should be able to grow with the business.
4) **Performance**: All the core elements of the data center should be able to provide optimal performance and service all processing requests at high speed. The infrastructure should be able to support performance requirements.

5) **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. Any variation in data during its retrieval implies corruption, which may affect the operations of the organization.
6) **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.
7) **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.

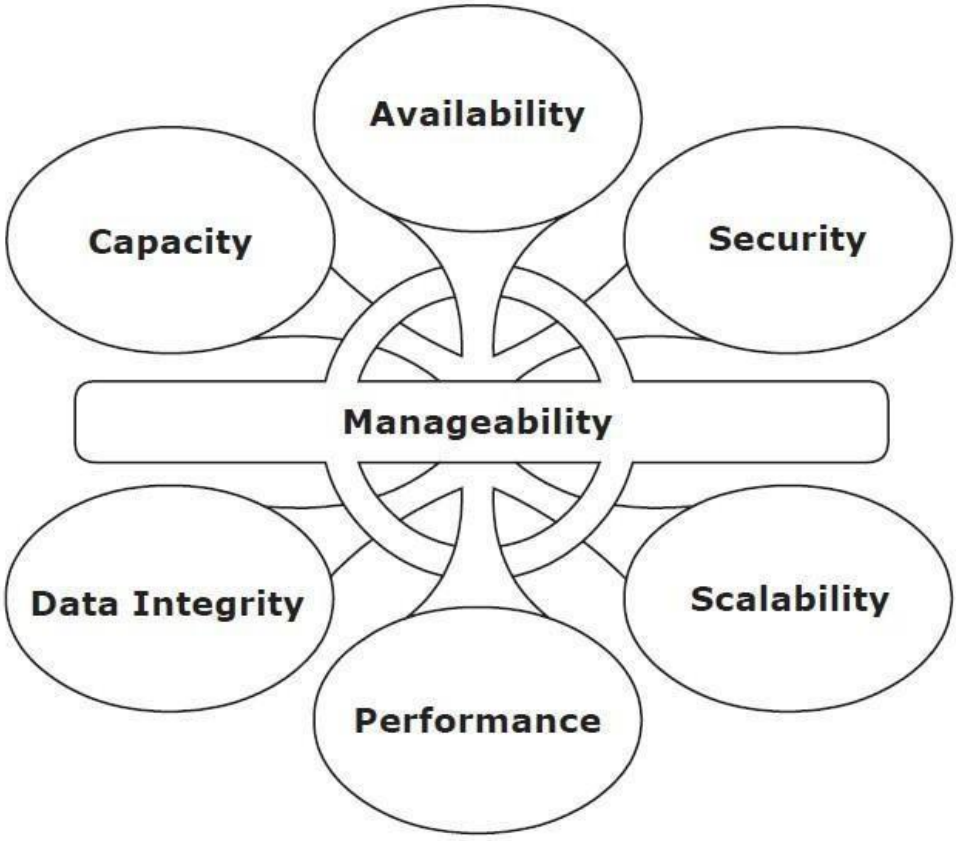


Fig 1.6: Key characteristics of data center elements

1.3.3 Managing a Data Center

- Managing a data-center involves many tasks.
- Key management-tasks are: 1) Monitoring 2) Reporting and 3) Provisioning.

1) Monitoring is a process of continuous
→ collection of information and
→ review of the entire storage infrastructure (called as Information Storage System).

- Following parameters are monitored:

i) Security
ii) Performance
iii) Accessibility and
iv) Capacity.

2) Reporting is done periodically on performance, capacity and utilization of the resources.

- Reporting tasks help to

→ establish business-justifications and
→ establish chargeback of costs associated with operations of data-center.

3) Provisioning is process of providing h/w, s/w & other resources needed to run a data-center.

- Main tasks are: i) Capacity Planning and ii) Resource Planning.

i) Capacity Planning
- It ensures that future needs of both user & application will be addressed in most cost-effective way

ii) Resource Planning
- It is the process of evaluating & identifying required resources such as

→ Personnel (employees)
→ Facility (site or plant) and
→ Technology (Artificial Intelligence, Deep Learning).

1.5 Virtualization

- Virtualization is a technique of abstracting physical resources, such as compute, storage, and network, and making them appear as logical resources.
- Virtualization has existed in the IT industry for several years and in different forms.
- Common examples of virtualization are virtual memory used on compute systems and partitioning of raw disks.
- Virtualization enables pooling of physical resources and providing an aggregated view of the physical resource capabilities. For example, storage virtualization enables multiple pooled storage devices to appear as a single large storage entity.
- Similarly, by using compute virtualization, the CPU capacity of the pooled physical servers can be viewed as the aggregation of the power of all CPUs (in megahertz).
- Virtualization also enables centralized management of pooled resources.
- Virtual resources can be created and provisioned from the pooled physical resources. For example, a virtual disk of a given capacity can be created from a storage pool or a virtual server with specific CPU power and memory can be configured from a compute pool.
- These virtual resources share pooled physical resources, which improves the utilization of physical IT resources.
- Based on business requirements, capacity can be added to or removed from the virtual resources without any disruption to applications or users.
- With improved utilization of IT assets, organizations save the costs associated management of new physical resources. Moreover, fewer physical resources means less space and energy, which leads to better economics and green computing.

1.6 Cloud Computing

- Cloud computing enables individuals or businesses to use IT resources as a service over the network.
- It provides highly scalable and flexible computing that enables provisioning of resources on demand.
- Users can scale up or scale down the demand of computing resources, including storage

capacity, with minimal management effort or service provider interaction.

- Cloud computing empowers self-service requesting through a fully automated request-fulfillment process.
- Cloud computing enables consumption-based metering; therefore, consumers pay only for the resources they use, such as CPU hours used, amount of data transferred, and gigabytes of data stored.
- Cloud infrastructure is usually built upon virtualized data centers, which provide resource pooling and rapid provisioning of resources.

1.7 Key Data center Elements

1.7.1 Application

- An application is a computer program that provides the logic for computing operations.
- The application sends requests to the underlying operating system to perform read/write (R/W) operations on the storage devices.
- Applications 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

1.7.2.DBMS

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

1.7.3 Host(or) Compute

- The computers on which applications run are referred to as hosts. Hosts can range from simple laptops to complex clusters of servers.
- Hosts can be physical or virtual machines.
- A compute virtualization software enables creating virtual machines on top of a physical

compute infrastructure.

- A host consists of
  - ✓ CPU: The CPU consists of four components-Arithmetic Logic Unit (ALU), control unit, registers, and L1 cache
  - ✓ Memory: There are two types of memory on a host, Random Access Memory (RAM) and Read-Only Memory (ROM)
  - ✓ I/O devices : keyboard, mouse, monitor
  - ✓ a collection of software to perform computing operations- This software includes the operating system, file system, logical volume manager, device drivers, and so on.

The following section details various software components that are essential parts of a host system.

1.7.3.1 Operating System

- In a traditional computing environment, an operating system controls all aspects of computing.
- It works between the application and the physical components of a compute system.
- In a virtualized compute environment, the virtualization layer works between the operating system and the hardware resources.

Functions of OS

- data access
- monitors and responds to user actions and the environment
- organizes and controls hardware components
- manages the allocation of hardware resources
- It provides basic security for the access and usage of all managed resources
- performs basic storage management tasks
- manages the file system, volume manager, and device drivers.

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 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 operating system utility that manages the virtual memory is known as the virtual memory manager (VMM).
- 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.

1.7.3.2 Device Drivers

- 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.

1.7.3.3 Volume Manager

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

Disadvantages:

- ✓ lack of flexibility.
- ✓ When a disk drive ran out of space, there was no easy way to extend the file system's size.
- ✓ 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

Solution: evolution of Logical Volume Managers (LVMs)

- LVM enabled dynamic extension of file system capacity and efficient storage management.

- The LVM is software that runs on the compute system and manages logical and physical storage.
- LVM is an intermediate layer between the file system and the physical disk.
- LVM can partition a larger-capacity disk into virtual, smaller-capacity volumes(called Partitioning) or aggregate several smaller disks to form a larger virtual volume. The process is called 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 Fig 1.7)

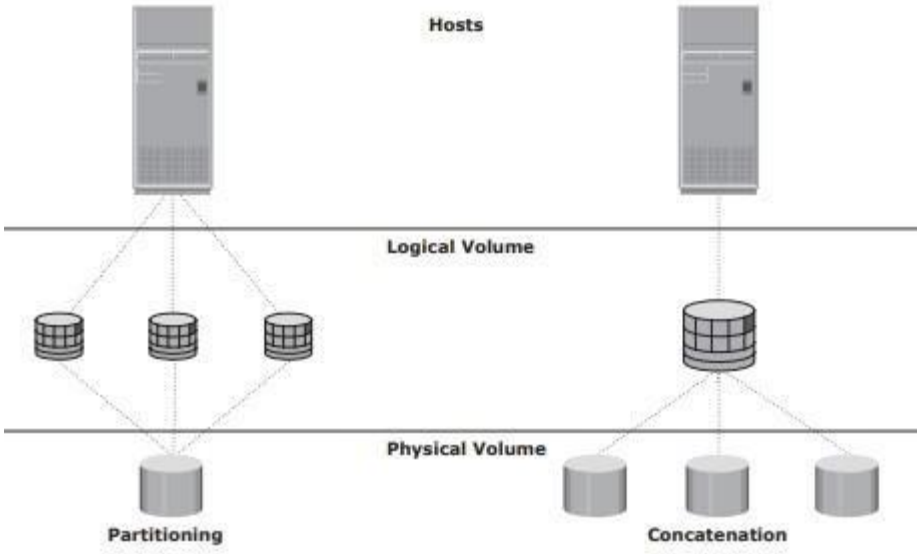


Fig 1.7: Disk Partitioning and concatenation

- Concatenation is the process of grouping several physical drives and presenting them to the host as one big logical volume.
- The basic LVM components are **physical volumes, volume groups, and logical volumes**.
- Each physical disk connected to the host system is a **physical volume (PV).**
- 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. 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.

1.7.3.4 File System

- A file is a **collection of related records** or data stored as a unit with a name.
- A file system is a hierarchical structure of files.
- A file system enables easy access to data files residing within a disk drive, a disk partition, or a logical volume.
- It provides users with the functionality to create, modify, delete, and access files.
- Access to files on the disks is controlled by the permissions assigned to the file by the owner, which are also maintained by the file system.
- A file system organizes data in a structured hierarchical manner via the use of directories, which are containers for storing pointers to multiple files.
- All file systems maintain a pointer map to the directories, subdirectories, and files that are part of the file system.
- Examples of common file systems are:
  - ✓ FAT 32 (File Allocation Table) for Microsoft Windows
  - ✓ NT File System (NTFS) for Microsoft Windows
  - ✓ UNIX File System (UFS) for UNIX
  - ✓ Extended File System (EXT2/3) for Linux
- 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.**
- A superblock contains important information about the file system, such as the file system type, creation and modification dates, size, and layout.
- 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.

- The following list shows the process of mapping user files to the disk storage subsystem with an LVM (see Fig 1.8)

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

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

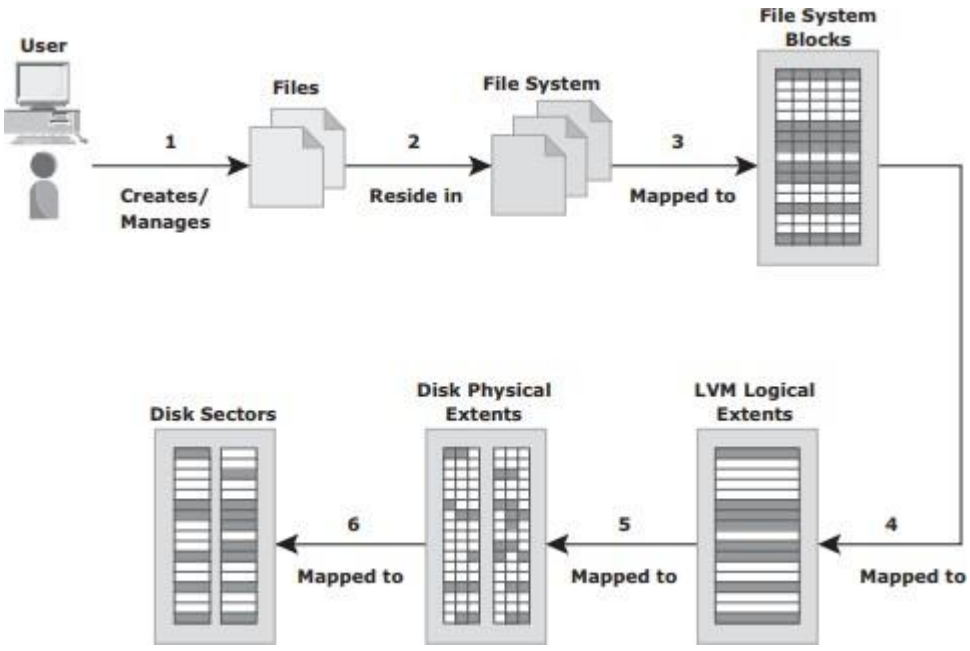


Fig 1.8: Process of mapping user files to disk storage

- The file system tree starts with the root directory. The root directory has a number of subdirectories.
- A file system can be either :
  - ✓ a journaling file system
  - ✓ a nonjournaling file system.

Nonjournaling file system : Nonjournaling file systems cause a potential loss of files because they use separate writes to update their data and metadata. If the system crashes during the write process, the metadata or data might be lost or corrupted. When the system reboots, the file system attempts to update the metadata structures by examining and repairing them. This operation takes a long time on large file systems. If there is insufficient information to re-create the wanted or original structure, the files might be misplaced or lost, resulting in corrupted file systems.

Journaling file system: Journaling File System uses a separate area called a *log* or *journal*. This journal might contain all the data to be written (physical journal) or just the metadata to be updated (logical journal). Before changes are made to the file system, they are written to this separate area. After the journal has been updated, the operation on the file system can be performed. If the system crashes during the operation, there is enough information in the log to "*replay*" the log record and complete the operation. Nearly all file system implementations today use journaling

Advantages:

- Journaling results in a quick file system check because it looks only at the active, most recently accessed parts of a large file system.
- Since information about the pending operation is saved, the risk of files being lost is reduced.

Disadvantage:

- they are slower than other file systems. This slowdown is the result of the extra operations that have to be performed on the journal each time the file system is changed.
- But the advantages of lesser time for file system checks and maintaining file system integrity far outweighs its disadvantage.

1.7.3.5 Compute Virtualization

- Compute virtualization is a technique for *masking* or *abstracting* the physical hardware from the operating system. It enables multiple operating systems to run concurrently on single or clustered physical machines.
- This technique enables creating portable virtual compute systems called *virtual machines* (VMs) running its own 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 called the *hypervisor*. The hypervisor provides hardware resources, such as CPU, memory, and network to all the virtual machines.

- 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.
- Before Compute virtualization:
  - ✓ A physical server often faces resource-conflict issues when two or more applications running on the same server have conflicting requirements. As a result, only one application can be run on a server at a time, as shown in Fig 1.9 (a).
  - ✓ Due to this, organizations will need to purchase new physical machines for every application they deploy, resulting in expensive and inflexible infrastructure.
  - ✓ Many applications do not fully utilize complete hardware capabilities available to them. Resources such as processors, memory and storage remain underutilized.
  - ✓ Compute virtualization enables users to overcome these challenges (see Fig 1.9 (b)).

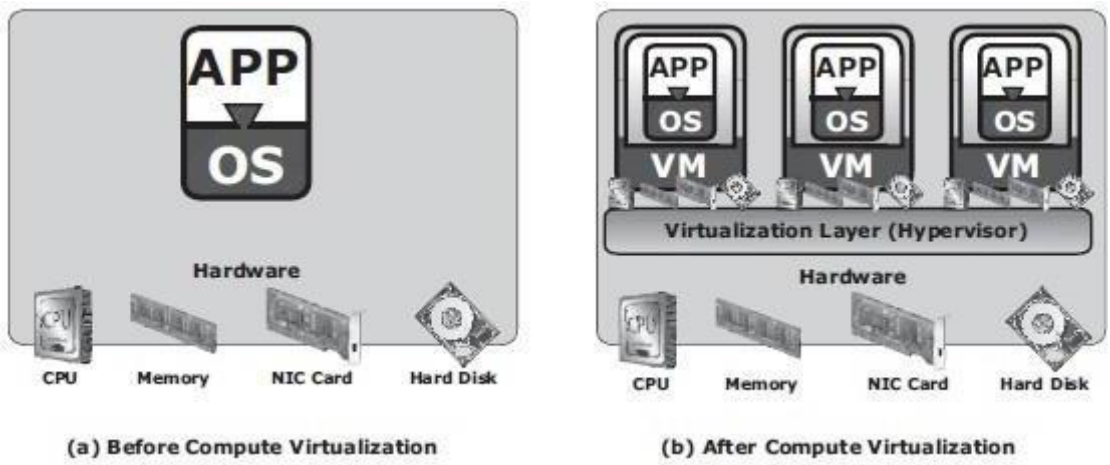


Fig 1.9: Server Virtualization

- After Compute virtualization:
  - ✓ This technique significantly improves server utilization and provides server consolidation.
  - ✓ *Server consolidation* enables organizations to run their data center with fewer physical servers.
  - ✓ This, in turn,

- reduces cost of new server acquisition,
- reduces operational cost,
- saves data center floor and rack space.

✓ Individual VMs can be restarted, upgraded, or even crashed, without affecting the other VMs.

✓ VMs can be copied or moved from one physical machine to another (non-disruptive migration) without causing application downtime. This is required for maintenance activities

1.8 Connectivity

➢ Connectivity refers to the interconnection between hosts or between a host and peripheral devices, such as printers or storage devices.

➢ Connectivity and communication between host and storage are enabled using:

✓ physical components

✓ interface protocols.

1.8.1 Physical Components of Connectivity

➢ The physical components of connectivity are the hardware elements that connect the host to storage.

➢ Three physical components of connectivity between the host and storage are (refer Fig 1.10):

✓ the host interface device

✓ port

✓ cable.

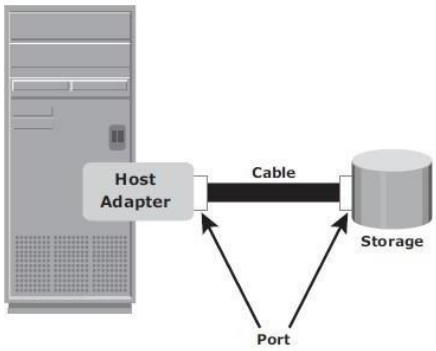


Fig 1.10: Physical components of connectivity

- A *host interface device* or *host adapter* connects a host to other hosts and storage devices.
  - ✓ Eg: host bus adapter (HBA) and network interface card (NIC).
  - ✓ HBA 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.
- *Cables* connect hosts to internal or external devices using copper or fiber optic media.

1.8.2 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:
  i. Integrated Device Electronics/Advanced Technology Attachment (IDE/ATA)
  ii. Small Computer System Interface (SCSI),
  iii. Fibre Channel (FC)
  iv. 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.
- If performance of the drive is important, sharing a port between two devices is not

recommended.

- The serial version of this protocol is known as Serial ATA (SATA) and supports single bit serial transmission.
- *High performance* and *low cost* SATA has 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.
- The high cost associated with SCSI limits its popularity among home or personal desktop users.
- SCSI supports up to 16 devices on a single bus and provides data transfer rates up to **640 MB/s.**
- **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 (FC):

- **Fibre Channel** is a widely used protocol for high-speed communication to the storage device.
- 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:
  - ✓ cost
  - ✓ maturity

- ✓ enables organizations to leverage their existing IP-based network.

- **iSCSI** and **FCIP** protocols are common examples that leverage IP for host-to-storage communication.

1.9 Storage

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

Tapes

- In the past, **tapes** were the most popular storage option for backups because of their low cost.
- Tapes have various limitations in terms of performance and management, as listed below:
  - i. Data is stored on the tape linearly along the length of the tape. Search and retrieval of data are done sequentially, and it invariably takes several seconds to access the data. As a result, **random data access is slow and time-consuming**.
  - ii. 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.
  - iii. On a tape drive, the read/write head touches the tape surface, so the tape degrades or wears out after repeated use.
  - iv. The storage and retrieval requirements of data from the tape and the overhead associated with managing the tape media are significant.
- Due to these limitations and availability of low-cost disk drives, tapes are no longer a preferred choice as a backup destination for enterprise-class data centers.

Optical Disc Storage:

- It is popular in small, single-user computing environments.
- It is frequently used by individuals to store photos or as a backup medium on personal or laptop computers.

- It is also used as a distribution medium for small applications, such as games, or as a means to transfer small amounts of data from one computer system to another.
- The capability to **write once and read many (WORM)** is one advantage of optical disc storage. Eg: CD-ROM
- Collections of optical discs in an array, called a **jukebox**, are still used as a fixed-content storage solution.
- Other forms of optical discs include CD-RW, Blu-ray disc, and other variations of DVD.

Disk Drives:

- **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.
- Disks have large capacity.
- Disk storage arrays are configured with multiple disks to provide **increased capacity** and **enhanced performance**.
- Disk drives are accessed through predefined protocols, such as ATA, SATA, SAS, and FC.
- These protocols are implemented on the disk interface controllers.
- Disk interface controllers were earlier implemented as separate cards, which were connected to the motherboard.
- Modern disk interface controllers are integrated with the disk drives; therefore, disk drives are known by the protocol interface they support, for example SATA disk, FC disk, etc.

Data Protection: RAID

- 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)**."
- **RAID is the use of small-capacity, inexpensive disk drives as an alternative to large-capacity drives common on mainframe computers.**
- Later RAID has been redefined to refer to ***independent*** disks to reflect advances in the storage technology.

- RAID stands for Redundant Array of Independent Disk.
- RAID is the way of combining several independent small disks into a single large-size storage.
- It appears to the OS as a single large-size disk.
- It is used to increase performance and availability of data-storage.
- There are two types of RAID implementation 1) hardware and 2) software.
- RAID-controller is a specialized hardware which

→ performs all RAID-calculations and

→ presents disk-volumes to host.

- Key functions of RAID-controllers:

1) Management and control of disk-aggregations.
2) Translation of I/O-requests between logical-disks and physical-disks.
3) Data-regeneration in case of disk-failures.

1.10 RAID Implementation Methods

- The two methods of RAID implementation are:
    1. Hardware RAID.
    2. Software RAID.

1.10.1 Hardware RAID

- In hardware RAID implementations, a specialized hardware controller is implemented either on the *host* or on the *array*.
- **Controller card RAID** is a *host-based hardware RAID* implementation in which a specialized RAID controller is installed in the host, and disk drives are connected to it.
- Manufacturers also integrate RAID controllers on motherboards.
- A host-based RAID controller is not an efficient solution in a data center environment with a large number of hosts.
- 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, and the host manages these volumes as physical drives.
- The key functions of the RAID controllers are as follows:
    - ✓ 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

1.10.2 Software RAID

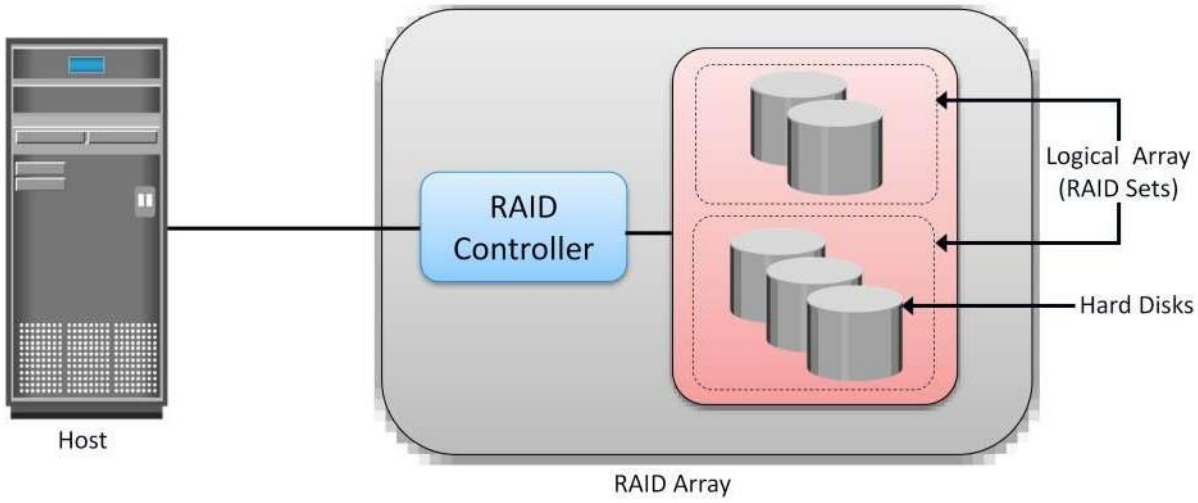
- **Software RAID** uses host-based software to provide RAID functions.
- It is implemented at the operating-system level and does not use a dedicated hardware controller to manage the RAID array.
- Advantages when compared to Hardware RAID:
    - ✓ cost
    - ✓ simplicity benefits

- Limitations:
  - ✓ **Performance**: Software RAID affects overall system performance. This is due to additional CPU cycles required to perform RAID calculations.
  - ✓ **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.

1.11 RAID Array Components

- A RAID array is an enclosure that contains a number of disk drives and supporting Hardware to implement RAID.
- A subset of disks within a RAID array can be grouped to form logical associations called logical arrays, also known as a RAID set or a RAID group

Fig: Components of RAID array



1.11 RAID Techniques

- There are three RAID techniques
    1. striping
    2. mirroring
    3. parity

1.11.1 Striping

- **Striping** is a technique to spread data across multiple drives (more than one) to use the drives in parallel.
- All the read-write heads work simultaneously, allowing more data to be processed in a shorter time and increasing performance, compared to reading and writing from a single disk.
- Within each disk in a RAID set, a **predefined number of contiguously addressable** disk blocks are defined as a **strip**.
- The set of aligned strips that spans across all the disks within the RAID set is called a **stripe**.
- Fig 1.11 shows physical and logical representations of a striped RAID set.

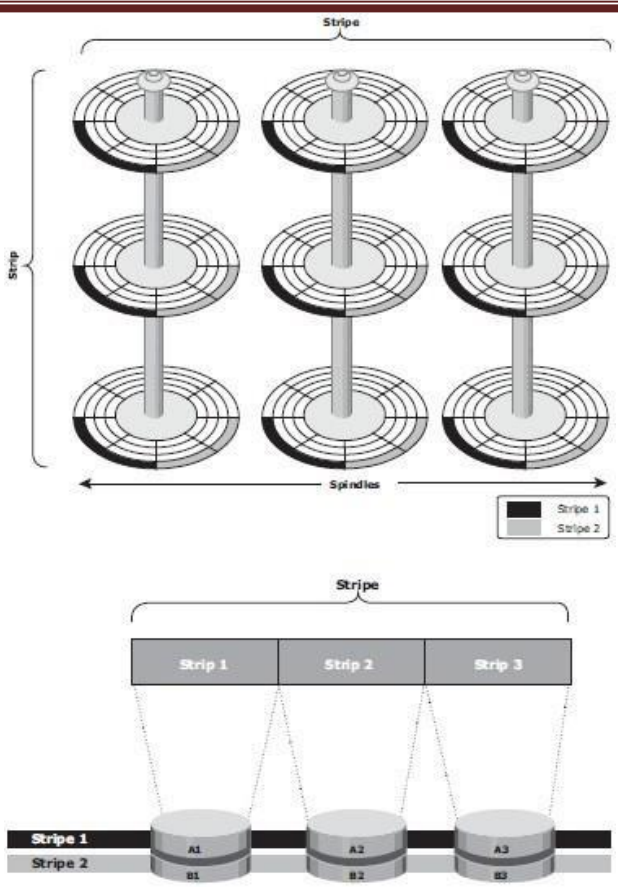


Fig 1.11: Striped RAID set

- **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 disk in the set.
- All strips in a stripe have the same number of blocks.
  - ✓ Having a smaller strip size means that data is broken into smaller pieces while spread across the disks.
- **Stripe size** is a multiple of strip size by the number of **data** disks in the RAID set.
  - ✓ Eg: In a 5 disk striped RAID set with a strip size of 64 KB, the stripe size is 320KB (64KB x 5).
- **Stripe width** refers to the number of *data* strips in a stripe.
- Striped RAID does not provide any data protection unless parity or mirroring is used.

1.11.2 Mirroring

- **Mirroring** is a technique whereby the same data is stored on two different disk drives, yielding two copies of the data.
- If one disk drive failure occurs, the data is intact on the surviving disk drive (see Fig 1.12) and the controller continues to service the host's data requests from the surviving disk of a mirrored pair.
- 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.

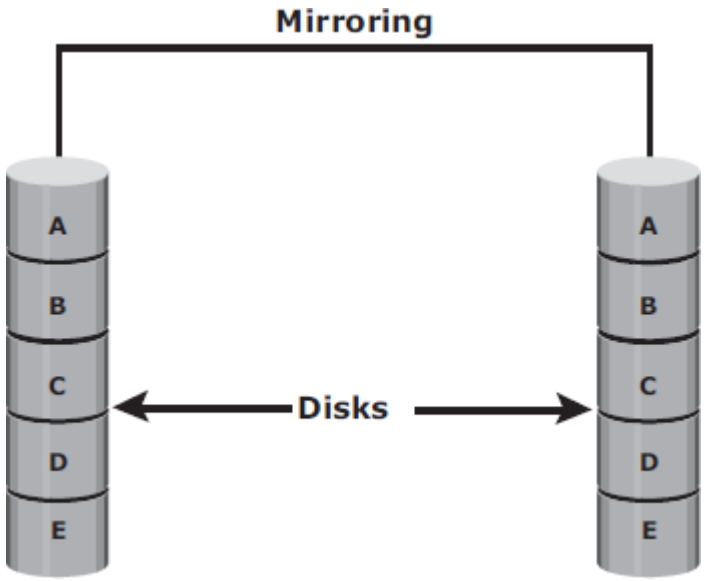


Fig 1.12: Mirrored disks in an array

- Advantages:
  - ✓ complete data redundancy,
  - ✓ mirroring enables fast recovery from disk failure.
  - ✓ data protection
- Mirroring is not a substitute for data backup. Mirroring constantly captures changes in the data, whereas a backup captures point-in-time images of the data.
- Disadvantages:
  - ✓ Mirroring involves duplication of data — the amount of storage capacity needed is

twice the amount of data being stored.

- ✓ Expensive

1.11.3 Parity

- **Parity** is a method to protect striped data from disk drive failure without the cost of mirroring.
- *An additional disk drive is added to hold parity*, a mathematical construct that allows re-creation of the missing data.
- Parity is a **redundancy technique** that ensures protection of data without maintaining a full set of duplicate data.
- Calculation of parity is a function of the RAID controller.
- Parity information can be stored on separate, dedicated disk drives or distributed across all the drives in a RAID set.
- Fig 1.13 shows a parity RAID set.

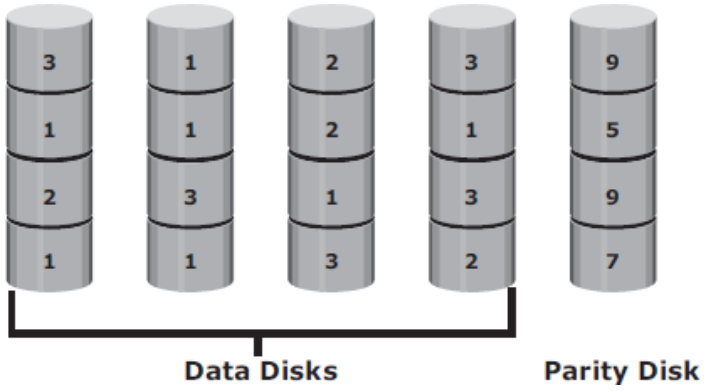


Fig 1.13: Parity RAID

- The first four disks, labeled "*Data Disks*," contain the data. The fifth disk, labeled "*Parity Disk*," stores the parity information, which, in this case, is the sum of the elements in each row.
- Now, if one of the data disks fails, the missing value can be calculated by subtracting the sum of the rest of the elements from the parity value.
- Here, computation of parity is represented as an arithmetic sum of the data. However, parity calculation is a bitwise XOR operation.

XOR Operation:

- A bit-by-bit Exclusive -OR (XOR) operation takes two bit patterns of equal length and performs the logical XOR operation on each pair of corresponding bits.
- The result in each position is 1 if the two bits are different, and 0 if they are the same.

- The truth table of the XOR operation is shown below (A and B denote inputs and C, the output the XOR operation).

Table 1.1: Truth table for XOR Operation

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

- If any of the data from A, B, or C is lost, it can be reproduced by performing an XOR operation on the remaining available data.
- Eg: if a disk containing all the data from A fails, the data can be regenerated by performing an XOR between B and C.
- Advantages:
  - ✓ Compared to mirroring, parity implementation considerably reduces the **cost** associated with data protection.
- Disadvantages:
  - ✓ 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 array.
- For parity RAID, the stripe size calculation does not include the parity strip.
- Eg: in a five (4 + 1) disk parity RAID set with a strip size of 64 KB, the stripe size will be 256 KB (64 KB x 4).

1.12 RAID Levels

- RAID Level selection is determined by below factors:
  - ✓ Application performance
  - ✓ data availability requirements
  - ✓ cost
- RAID Levels are defined on the basis of:
  - ✓ Striping
  - ✓ Mirroring
  - ✓ Parity techniques
- Some RAID levels use a single technique whereas others use a combination of techniques.
- Table 1.2 shows the commonly used RAID levels

Table 1.2: RAID Levels

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

1.12.1 RAID 0

- **RAID 0** configuration uses *data striping techniques*, where data is striped across all the disks within a RAID set. Therefore it utilizes the full storage capacity of a RAID set.
- To read data, all the strips are put back together by the controller.
- Fig 1.14 shows RAID 0 in an array in which data is striped across five disks.

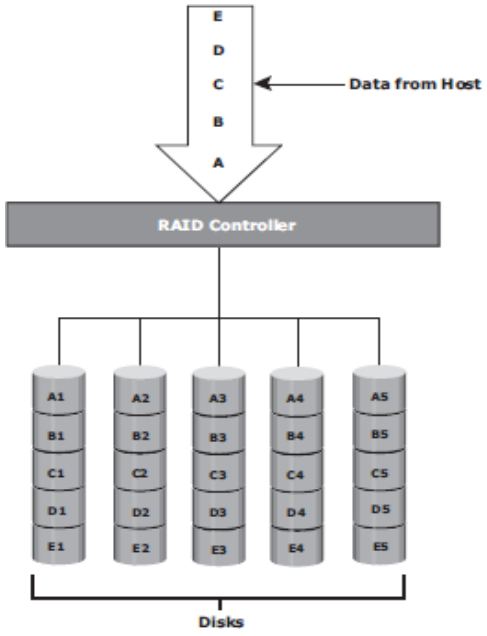


Fig 1.14: RAID 0

- When the number of drives in the RAID set increases, performance improves because more data can be read or written simultaneously.
- RAID 0 is a good option for applications that need high I/O throughput.
- However, if these applications require high availability during drive failures, RAID 0 does not provide data protection and availability.

1.12.2 RAID 1

- **RAID 1** is based on the *mirroring* technique.
- In this RAID configuration, data is mirrored to provide *fault tolerance* (see Fig 1.15). A
- RAID 1 set consists of two disk drives and every write is written to both disks.
- The mirroring is transparent to the host.
- During disk failure, the impact on data recovery in RAID 1 is the least among all RAID implementations. This is because the RAID controller uses the mirror drive for data recovery.
- RAID 1 is suitable for applications that require high availability and cost is no constraint.

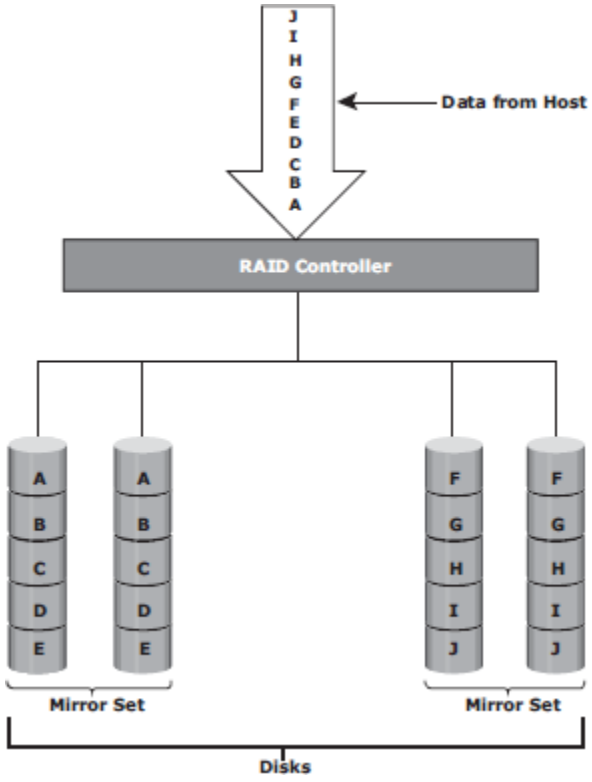


Fig 1.15: RAID 1

1.12.3 Nested RAID

- Most data centers require data redundancy and performance from their RAID arrays.
- RAID 1+0 and RAID 0+1 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 Fig 1.16).

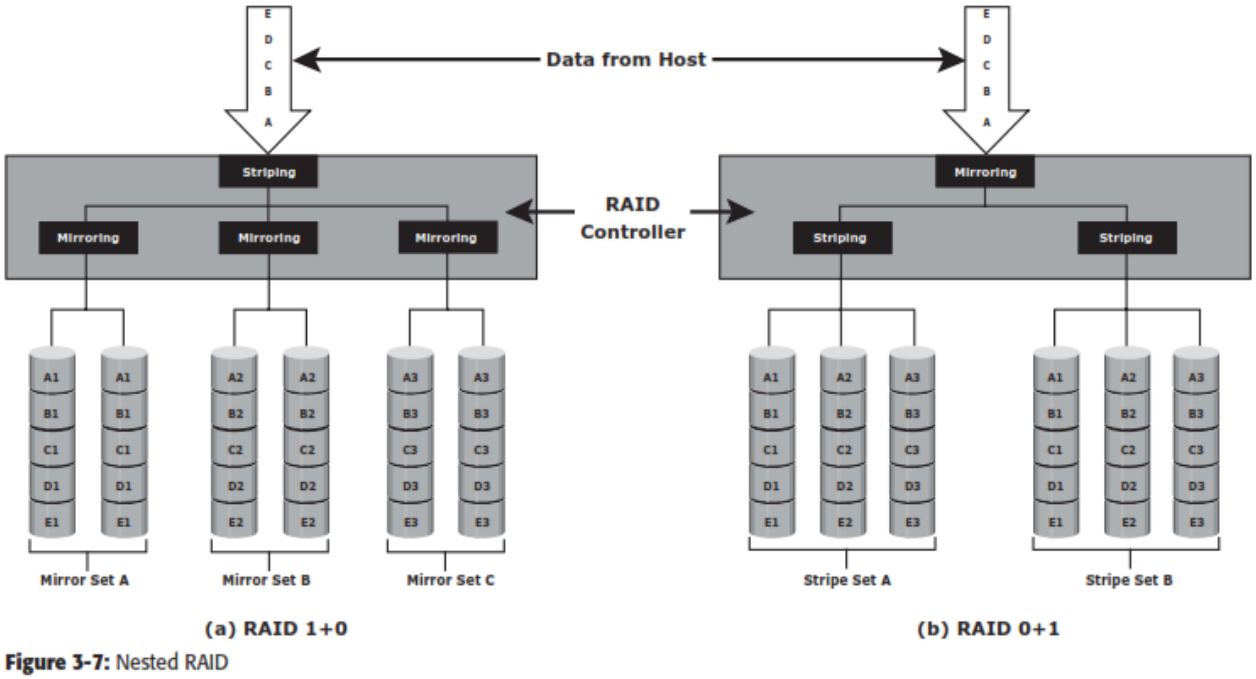


Figure 3-7: Nested RAID

Fig 1.16: Nested RAID

RAID 1+0:

- RAID 1+0 is also known as RAID 10 (Ten) or RAID 1/0.
- RAID 1+0 performs well for workloads with small, random, write-intensive I/Os.
- Some applications that benefit from RAID 1+0 include the following:
  - ✓ High transaction rate Online Transaction Processing (OLTP)
  - ✓ Large messaging installations
  - ✓ Database applications with write intensive random access workloads
- **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 the data are striped across multiple disk drive pairs in a RAID set.
- When replacing a failed drive, only the mirror is rebuilt. The disk array controller uses the surviving drive in the mirrored pair for data recovery and continuous operation.

Working of RAID 1+0:

- Eg: consider an example of six disks forming a RAID 1+0 (RAID 1 first and then RAID 0) set.
- These six disks are paired into three sets of two disks, where each set acts as a RAID 1 set (mirrored pair of disks). Data is then striped across all the three mirrored sets to form RAID 0.

- Following are the steps performed in RAID 1+0 (see Fig 1.16 [a]):
  - ✓ Drives 1+2 = RAID 1 (Mirror Set A)
  - ✓ Drives 3+4 = RAID 1 (Mirror Set B)
  - ✓ Drives 5+6 = RAID 1 (Mirror Set C)
- Now, RAID 0 striping is performed across sets A through C.
- In this configuration, if drive 5 fails, then the mirror set C alone is affected. It still has drive 6 and continues to function and the entire RAID 1+0 array also keeps functioning.
- Now, suppose drive 3 fails while drive 5 was being replaced. In this case the array still continues to function because drive 3 is in a different mirror set.
- So, in this configuration, up to three drives can fail without affecting the array, as long as they are all in different mirror sets.
- **RAID 0+1** is also called a mirrored stripe.
- The basic element of RAID 0+1 is a stripe. This means that the process of striping data across disk drives is performed initially, and then the entire stripe is mirrored.
- In this configuration if one drive fails, then the entire stripe is faulted.

  Working of RAID 0+1:
- Eg: Consider the same example of six disks forming a RAID 0+1 (that is, RAID 0 first and then RAID 1).
- Here, six disks are paired into two sets of three disks each.
- Each of these sets, in turn, act as a RAID 0 set that contains three disks and then these two sets are mirrored to form RAID 1.
- Following are the steps performed in RAID 0+1 (see Fig 1.16 [b]):
  - ✓ Drives 1 + 2 + 3 = RAID 0 (Stripe Set A)
  - ✓ Drives 4 + 5 + 6 = RAID 0 (Stripe Set B)
- These two stripe sets are mirrored.
- If one of the drives, say drive 3, fails, the entire stripe set A fails.
- A rebuild operation copies the entire stripe, copying the 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.

1.12.4 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.
- 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.
- 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.
- Fig 1.17 shows the RAID 3 implementation

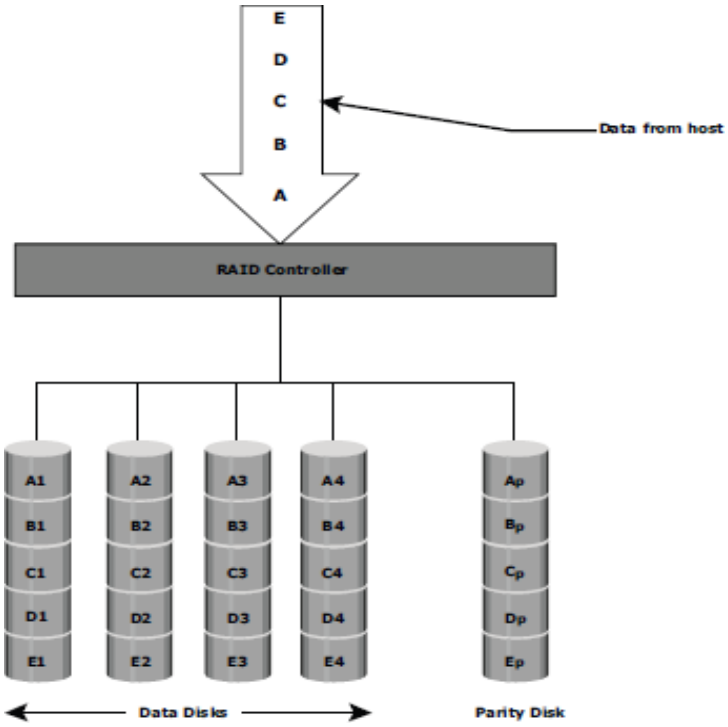


Fig 1.17: RAID 3

1.12.5 RAID 4

- RAID 4 stripes data for high performance and uses parity for improved fault tolerance. 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.

1.12.6 RAID 5

- RAID 5 is a versatile RAID implementation.
- It is similar to RAID 4 because it uses striping. The drives (strips) are also 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. Below Figure illustrates the RAID 5 implementation.
- Fig 1.18 illustrates the RAID 5 implementation.
- RAID 5 is good for random, read-intensive I/O applications and preferred for messaging, data mining, medium-performance media serving, and relational database management system (RDBMS) implementations, in which database administrators (DBAs) optimize data access.

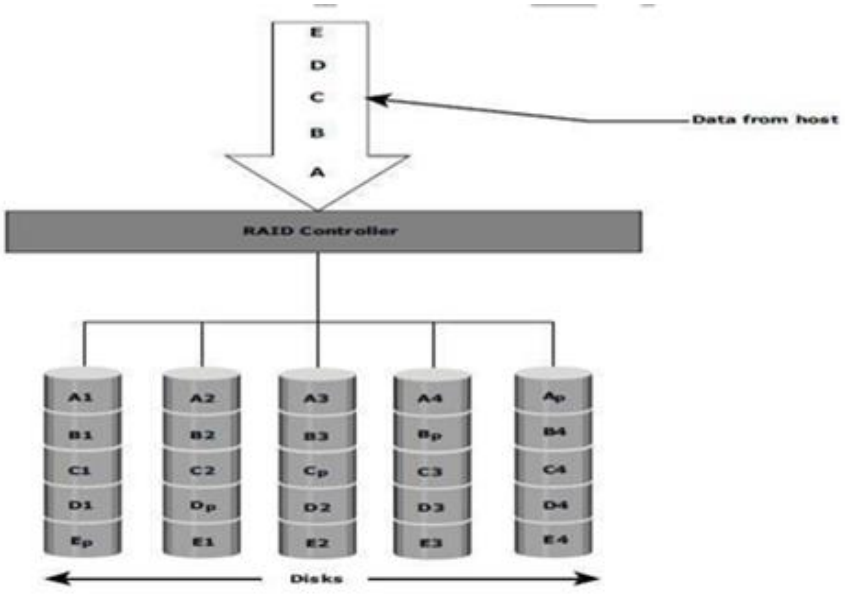


Fig 1.18: RAID 5

1.12.7 RAID 6

- RAID 6 includes a second parity element to enable survival in the event of the failure of two disks in a RAID group. 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.
- Fig 1.19 illustrates the RAID 6 implementation

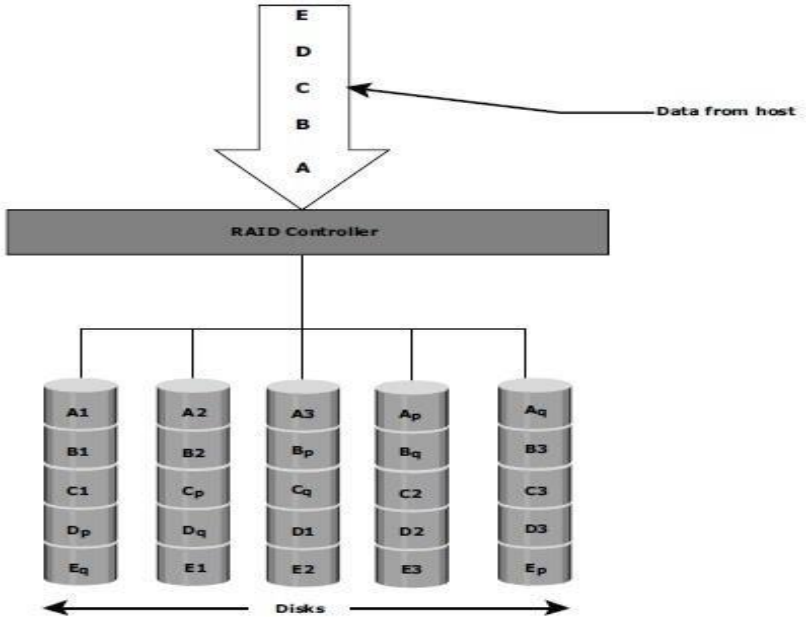


Fig 1.19: RAID 6

1.13 RAID Impact on Disk Performance

- When choosing a RAID type, it is imperative to consider its impact on disk performance and application IOPS.
- In both mirrored (RAID 1) and parity RAID (RAID 5) 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. **The write penalty is 2.**
- 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.
- Fig 1.20 illustrates a single write operation on RAID 5 that contains a group of five disks.

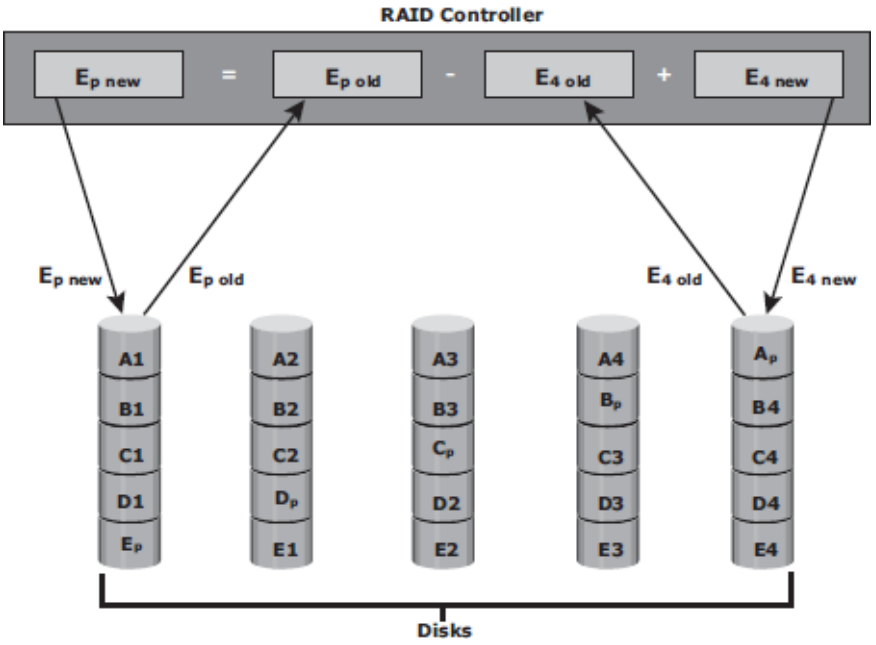


Fig 1.20: Write Penalty in RAID 5

- Four of these disks are used for data and one is used for parity.
- The **parity (E_p)** at the controller is calculated as follows:

$$E_p = E_1 + E_2 + E_3 + E_4 \text{ (XOR operations)}$$

- Whenever the controller performs a write I/O, parity must be computed by reading the old parity (E_p old) and the old data (E_4 old) from the disk, which means two read I/Os.
- The new parity (E_p new) is computed as follows:

$$E_p \text{ new} = E_p \text{ old} - E_4 \text{ old} + E_4 \text{ new (XOR operations)}$$

- After computing the new parity, the controller completes the write I/O by doing two write I/Os for the new data and the new parity onto the disks..
- Therefore, the controller performs two disk reads and two disk writes for every write operation, and **the write penalty is 4**.
- In RAID 6, which maintains dual parity, a disk write requires **three read operations**: two parity and one data.
- After calculating both new parities, the controller performs **three write operations**: two parity and an I/O.
- Therefore, in a RAID 6 implementation, the controller performs six I/O operations for each write I/O, and the **write penalty is 6**.

3.1.1 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.

i. 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 [1.0-0.6=0.4]
= 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

ii. 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 as follows:

RAID 5: 11,440 / 180 = 64 disks

RAID 1: 7,280 / 180 = 42 disks (approximated to the nearest even number)

Calculating IOPS from disks available:

Consider a server/storage with 8 450GB 15,000 RPM drives. We will consider two scenarios of Workload 80% Write 20%Read and another scenario with 20% Write 80% Read. Also we will calculate IOPS that cab be achieved in RAID5 and RAID 10 Scenario.

Total Raw IOPS = Disk Speed IOPS * Number of disks

Functional IOPS =(((Total Raw IOPS × Write %))/(RAID Penalty))+(Total Raw IOPS×Read %)

In our example ,

Total Raw IOPS = 175*8 = 1400 IOPS (Since 15K RPM disk can give 175 IOPS)

RAID-5:

Scenario 1(80% Write 20%Read) Functional IOPS = (((1400*0.8))/(4))+(1400*0.2) = 560 IOPS **Scenario 2**(20% Write 80%Read) Functional IOPS = (((1400*0.2))/(4))+(1400*0.8) = 1190 IOPS

RAID-1:

Scenario 1(80%Write20%Read) Functional IOPS = (((1400*0.8))/(2))+(1400*0.2) = 840 IOPS **Scenario 2**(20%Write80%Read) Functional IOPS = (((1400*0.2))/(2))+(1400*0.8) = 1260 IOPS

Calculating number of Disks required to achieve certain IOPS:

Consider a scenario where you will have to decide on RAID and number of disks required to achieve 2000 IOPS with a workload characterization of 80% Write 20%Read and another scenario with 20% Write 80% Read.

Total number of Disks required = ((Total Read IOPS + (Total Write IOPS*RAID Penalty))/Disk Speed IOPS)

Total IOPS = 2000

Note : 80% 0f 2000 IOPS = 1600 IOPS & 20% of 2000 IOPS = 400 IOPS

RAID-5:

Scenario 1(80% Write 20% Read) – Total Number of disks required = ((400+(1600*4))/175) = 39 Disks approximately

Scenario 2(20% Write 80%Read) – Total Number of disks required = ((1600+(400*4))/175) = 18 Disks approximately

RAID-1:

Scenario 1(80% Write 20% Read) – Total Number of disks required = ((400+(1600*2))/175) = 21 Disks approximately

Scenario 2(20% Write 80% Read) – Total Number of disks required = ((1600+(400*2))/175) = 14 Disks approximately

RAID Comparison

Table 3-2: Comparison of Common RAID Types

RAID	MIN. DISKS	STORAGE EFFICIENCY %	COST	READ PERFORMANCE	WRITE PERFORMANCE	WRITE PENALTY	PROTECTION
0	2	100	Low	Good for both random and sequential reads	Good	No	No protection
1	2	50	High	Better than single disk	Slower than single disk because every write must be committed to all disks	Moderate	Mirror protection
3	3	[(n-1)/n] × 100 where n= number of disks	Moderate	Fair for random reads and good for sequential reads	Poor to fair for small random writes and fair for large, sequential writes	High	Parity protection for single disk failure
4	3	[(n-1)/n] × 100 where n= number of disks	Moderate	Good for random and sequential reads	Fair for random and sequential writes	High	Parity protection for single disk failure
5	3	[(n-1)/n] × 100 where n= number of disks	Moderate	Good for random and sequential reads	Fair for random and sequential writes	High	Parity protection for single disk failure
6	4	[(n-2)/n] × 100 where n= number of disks	Moderate but more than RAID 5.	Good for random and sequential reads	Poor to fair for random writes and fair for sequential writes	Very High	Parity protection for two disk failures
1+0 and 0+1	4	50	High	Good	Good	Moderate	Mirror protection

1.14 Components of an Intelligent Storage System

- Intelligent Storage Systems are **feature-rich RAID arrays** that provide highly optimized I/O processing capabilities.
- These storage systems are configured with a large amount of memory (called *cache*) and multiple I/O paths and use sophisticated algorithms to meet the requirements of performance-sensitive applications.
- An intelligent storage system consists of **four key components** (Refer Fig 1.21):

  - ✓ Front End
  - ✓ Cache
  - ✓ Back end
  - ✓ Physical disks.
  - 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.
  - In modern intelligent storage systems, front end, cache, and back end are typically integrated on a single board (referred to as a storage processor or storage controller).

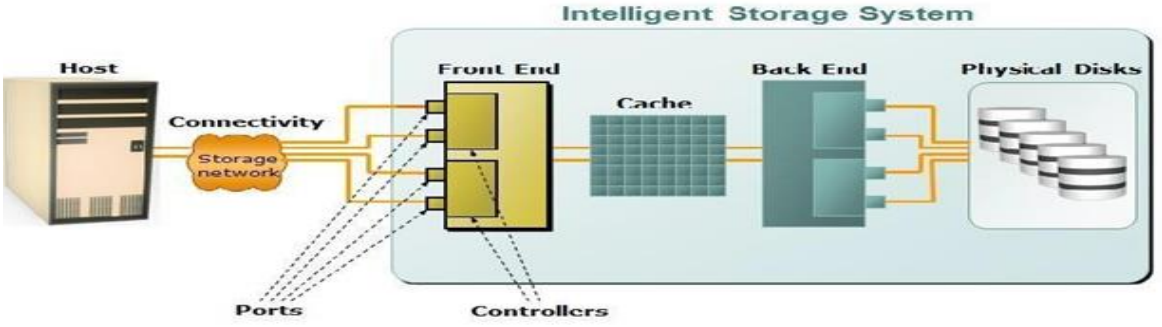


Fig 1.21 Components of an Intelligent Storage System

1.14.1 Front End

- The front end provides the interface between the storage system and the host.
- It consists of two components:
  i. Front-End Ports
  ii. Front-End Controllers.

- A front end has redundant controllers for high availability, and each controller contains multiple **front-end ports** that enable large numbers of hosts to connect to the intelligent storage system.
- Each front-end controller has processing logic that executes the appropriate transport protocol, such as Fibre Channel, iSCSI, FICON, or FCoE for storage connections.
- **Front-end controllers** route data to and from cache via the internal data bus.
- When the cache receives the write data, the controller sends an acknowledgment message back to the host.

1.14.2 Cache

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

Structure Of Cache

- Cache is organized into pages, which is the smallest unit of cache allocation. The size of a cache page is configured according to the application I/O size.
- Cache consists of the **data store** and **tag RAM**.
- The data store holds the data whereas the tag RAM tracks the location of the data in the data store (see Fig 1.22) and in the 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.
- 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.

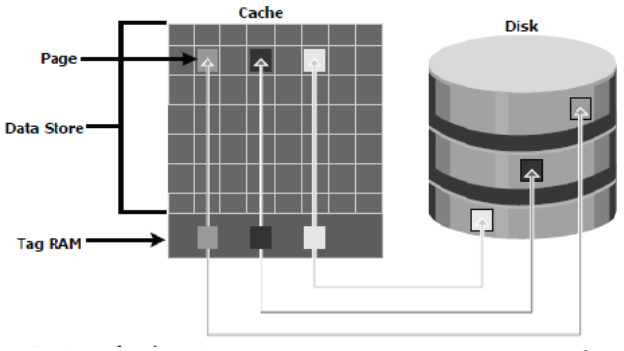


Fig 1.22: Structure of cache

Read Operation with Cache

- When a host issues a read request, the storage controller reads the tag RAM to determine whether the required data is available in cache.
- If the requested data is found in the cache, it is called a **read cache hit** or **read hit** and data is sent directly to the host, without any disk operation (see Fig 1.23[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. 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 pre-fetches an amount of data in multiples of the size of the host request.

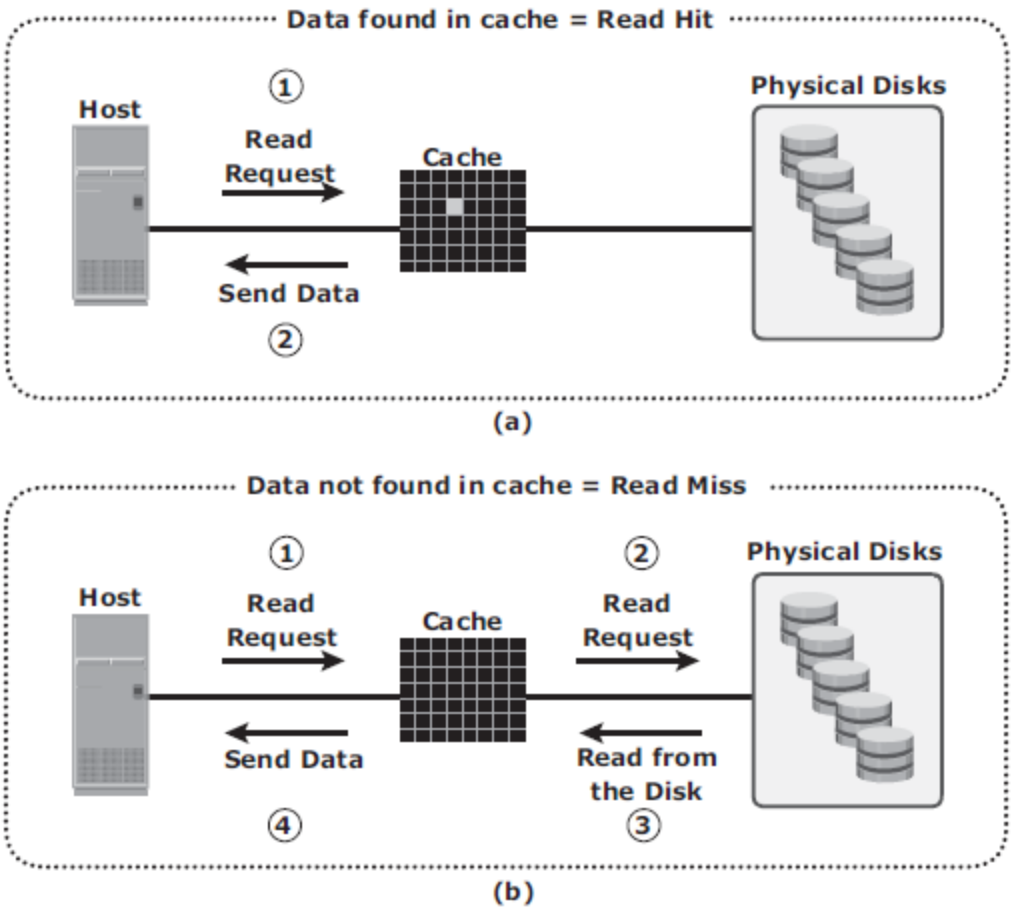


Fig 1.23 : Read hit and read miss

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

Cache Implementation

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

Cache Management

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

- ✓ **Most Recently Used (MRU):** 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) to manage space availability.
- **Flushing** is the process that commits data from cache to the disk.
- On the basis of the I/O access rate and pattern, high and low levels called **watermarks** are set in cache to manage the flushing process.
- **High watermark (HWM)** is the cache utilization level at which the storage system starts high-speed flushing of cache data.
- **Low watermark (LWM)** is the point at which the storage system stops flushing data to the disks.
- The *cache utilization level*, as shown in Fig 1.24, drives the mode of flushing to be used:
  - ✓ **Idle flushing:** Occurs continuously, at a modest rate, when the cache utilization level is between the high and low watermark.
  - ✓ **High watermark flushing:** Activated when cache utilization hits the high watermark. The storage system dedicates some additional resources for flushing. This type of flushing has some impact on I/O processing.
  - ✓ **Forced flushing:** Occurs in the event of a large I/O burst when cache reaches 100 percent of its capacity, which significantly affects the I/O response time. In forced flushing, system flushes the cache on priority by allocating more resources.

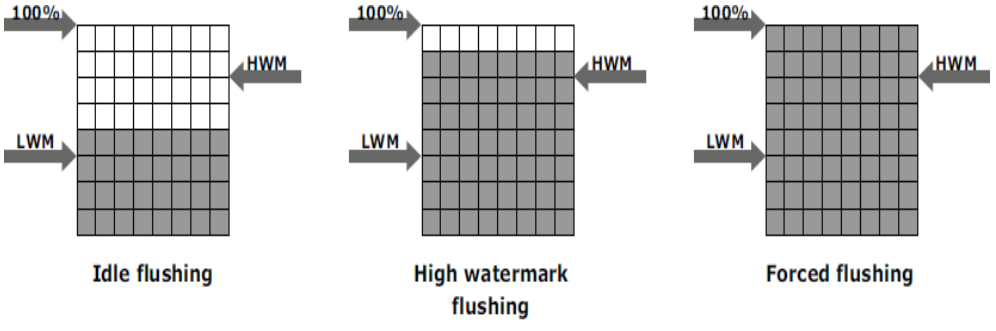


Fig 1.24 : Types of flushing

Cache Data Protection

- Cache is volatile memory, so a power failure or any kind of cache failure will cause loss of the data that is not yet committed to the disk.

- This risk of losing uncommitted data held in cache can be mitigated using
  i. cache mirroring
  ii. cache vaulting
- **Cache mirroring**
  - ✓ 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.
  - ✓ In cache mirroring approaches, the problem of maintaining *cache coherency* is introduced.
  - ✓ Cache coherency means that data in two different cache locations must be identical at all times. It is the responsibility of the array operating environment to ensure coherency.
- **Cache vaulting**
  - ✓ The risk of data loss due to power failure can be addressed in various ways:
    - powering the memory with a battery until the AC power is restored
    - using battery power to write the cache content to the disk.
  - ✓ If an extended power failure occurs, using batteries is not a viable option.
  - ✓ This is because in intelligent storage systems, large amounts of data might need to be committed to numerous disks, and batteries might not provide power for sufficient time to write each piece of data to its intended disk.
  - ✓ 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.

1.14.3 Back End

- The **back end** provides an interface between cache and the physical disks.
- It consists of two components:
  i. Back-end ports
  ii. 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, and also RAID functionality.
- For high data protection and high availability, storage systems are configured with dual controllers with multiple ports.

1.14.4 Physical Disk

- A physical disk stores data persistently.
- 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.

1.15 Types/ Implementation of Intelligent Storage Systems

- An intelligent storage system is divided into following two categories:
    1. High-end storage systems
    2. Midrange storage systems
- High-end storage systems have been implemented with active-active configuration, whereas midrange storage systems have been implemented with active-passive configuration.
- The distinctions between these two implementations are becoming increasingly insignificant.

1.15.1 High-end Storage Systems

- High-end storage systems, referred to as **active-active arrays,** are generally aimed at large enterprises for centralizing corporate data. These arrays are designed with a large number of controllers and cache memory.
- An active-active array implies that the host can perform I/Os to its LUNs across any of the available paths (see Fig 1.25).

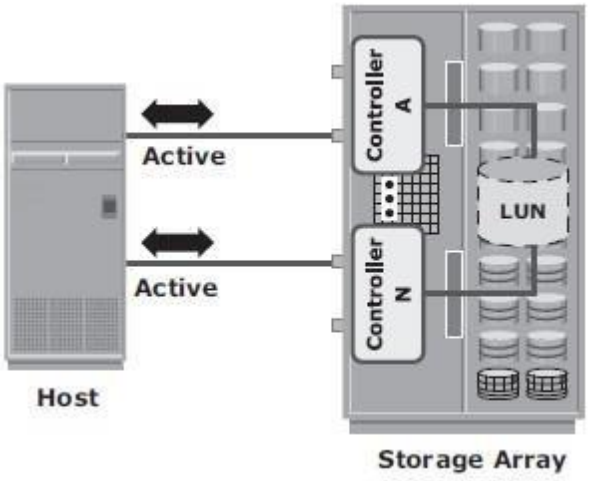


Fig 1.25 : Active-active configuration

Advantages of High-end storage:

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

1.15.2 Midrange Storage System

- Midrange storage systems are also referred to as **Active-Passive Arrays** and they are best suited for small- and medium-sized enterprises.
- They also provide optimal storage solutions at a *lower cost*.
- In an *active-passive* array, a host can perform I/Os to a LUN only through the paths to the **owning controller** of that LUN. These paths are called *Active Paths*. The other paths are *passive* with respect to this LUN.

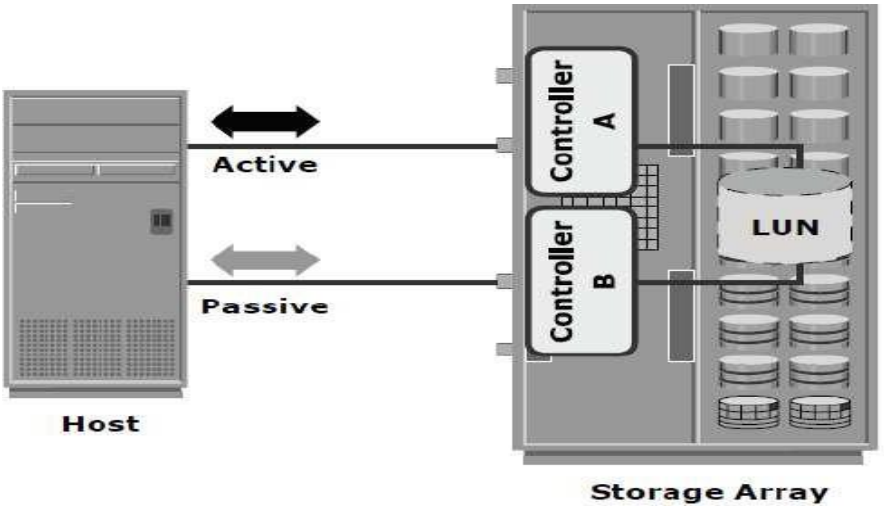


Fig 1.26 : Active-passive configuration

- As shown in Fig 1.26, the host can perform reads or writes to the LUN only through the path to controller A, as controller A is the owner of that LUN.
- The path to controller B remains **Passive** and no I/O activity is performed through this path.
- Midrange storage systems are typically designed with two controllers, each of which contains host interfaces, cache, RAID controllers, and disk drive interfaces.
- Midrange arrays are designed to meet the requirements of small and medium enterprise applications; therefore, they host less storage capacity and cache than high-end storage arrays.
- There are also fewer front-end ports for connection to hosts.
- But they ensure high redundancy and high performance for applications with predictable workloads.
- They also support array-based local and remote replication.

1.16 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 non-disruptively 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.
- Fig 1.27 illustrates the provisioning of thin LUNs.

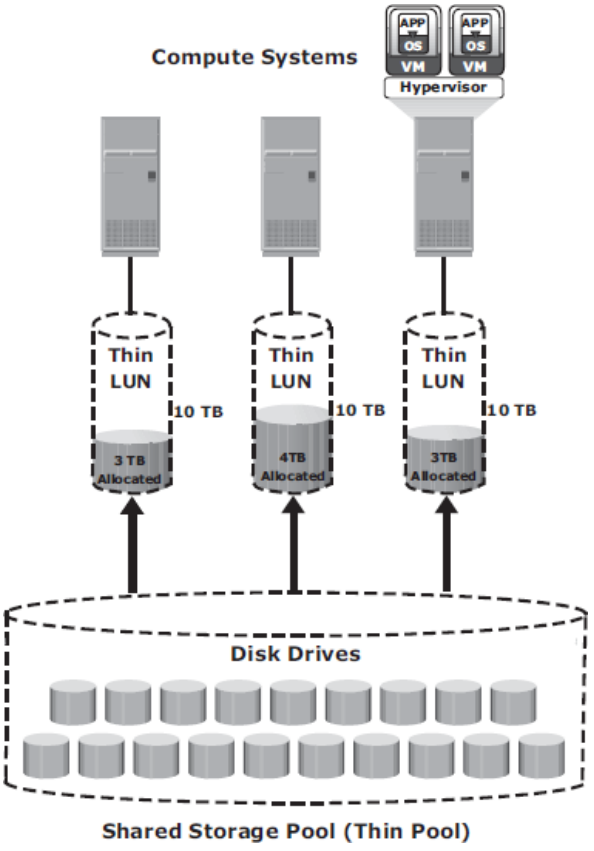


Fig 1.27: Virtual Provisioning

Comparison between Virtual and Traditional Storage Provisioning

- Virtual provisioning improves storage capacity utilization and simplifies storage management.
- Figure 1.28 shows an example, comparing virtual provisioning with traditional storage provisioning.

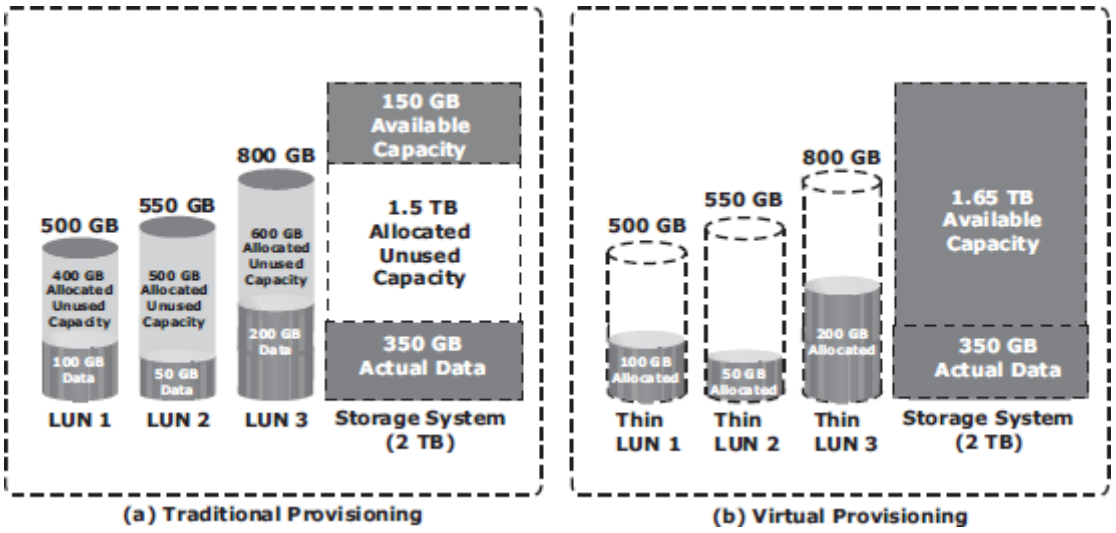


Fig 1.28: Traditional versus Virtual Provisioning

- With **traditional provisioning,** three LUNs are created and presented to one or more hosts (see Fig 1.28 [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 Fig 1.28 [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.**