Module 1

1 What is a data center? List the core components of data center. Explain the characteristics of data center. (08 Marks)

A data center is a facility that contains information storage and other physical information technology (IT) resources for computing, networking, and storing information. 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).

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.

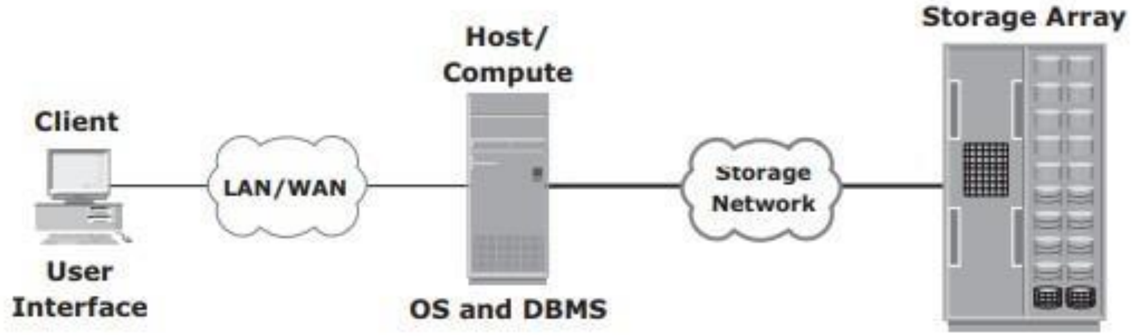
2) **Database**: More commonly, a database management system (DBMS) provides a structured way to store data in logically organized tables that are interrelated. A DBMS optimizes the storage and retrieval of data.

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.

Fig 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.



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.

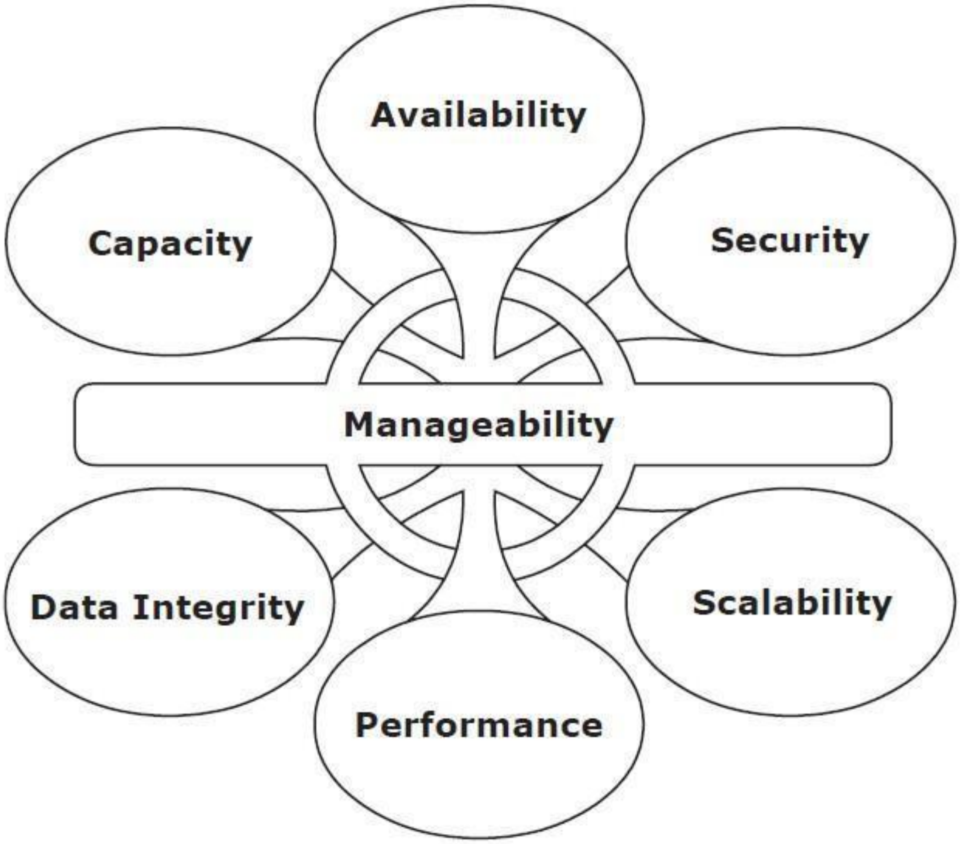
4) **Performance**: All the core elements of the data center should be able to provide optimal performance and service all processing requests at high speed. The infrastructure should be able to support performance requirements.

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.



2 Discuss volume manager and compute virtualization in detail.

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.

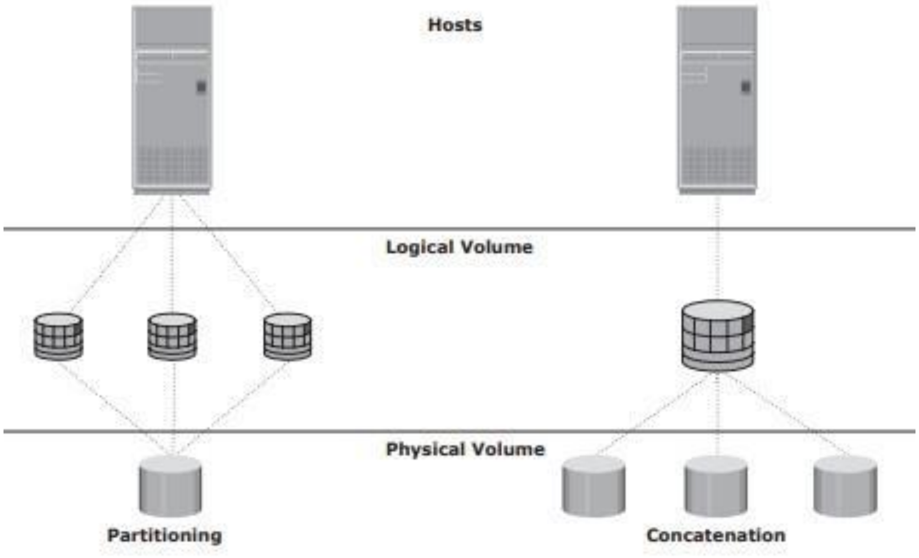
➢ The LVM is software that runs on the compute system and manages logical and physical storage.

➢ 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)



➢ 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.

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.

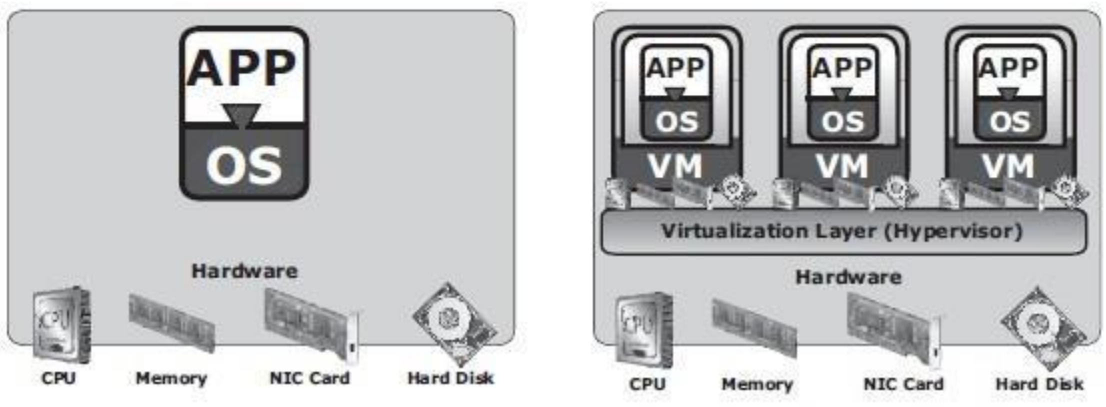
➢ **Before Compute virtualization:**

✓ 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)).



(a) Before Compute Virtualization (b) After Compute 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

3 Differentiate between software and hardware RAID. Illustrate how parity method is used for RAID levels.

➢ The two methods of RAID implementation are:

1. Hardware RAID.

2. Software RAID.

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

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.

Parity

➢ Parity is a method to protect striped data from disk drive failure without the cost of mirroring.

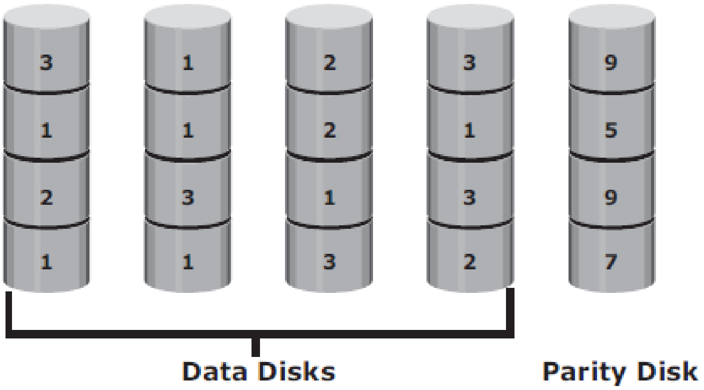
➢ An additional disk drive is added to hold parity, a mathematical construct that allows re-creation of the missing data.

➢ 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.



➢ 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).

4 With a neat diagram explain ISS. Explain in detail the cache components of ISS.

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

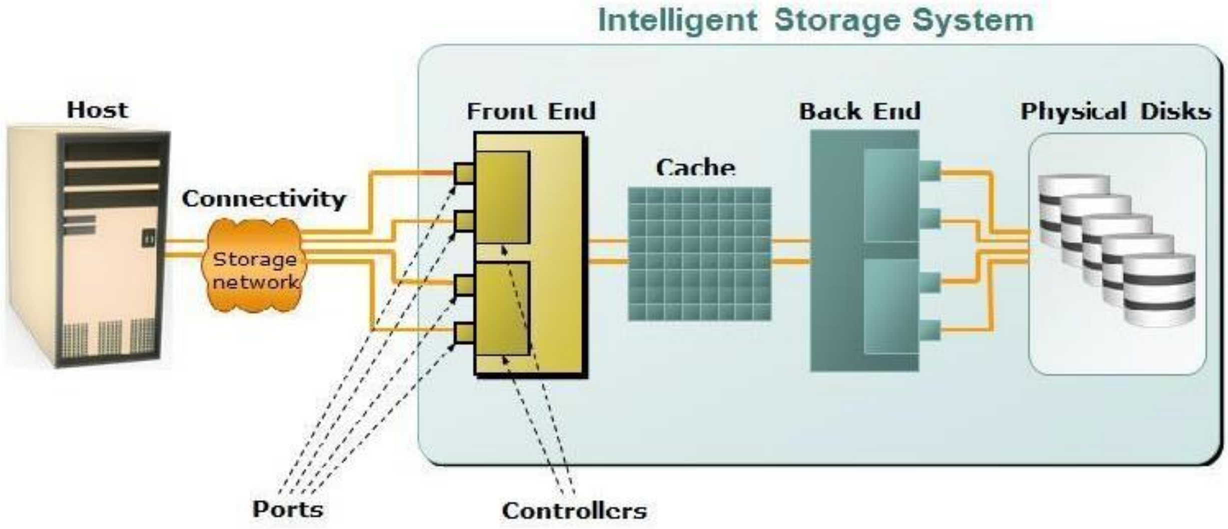
✓ Back end

✓ 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).



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 milliseconds 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.

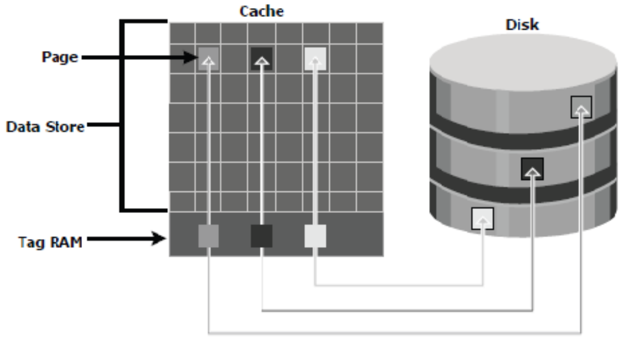
➢ Cache consists of the data store and tag RAM.

➢ 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.



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.

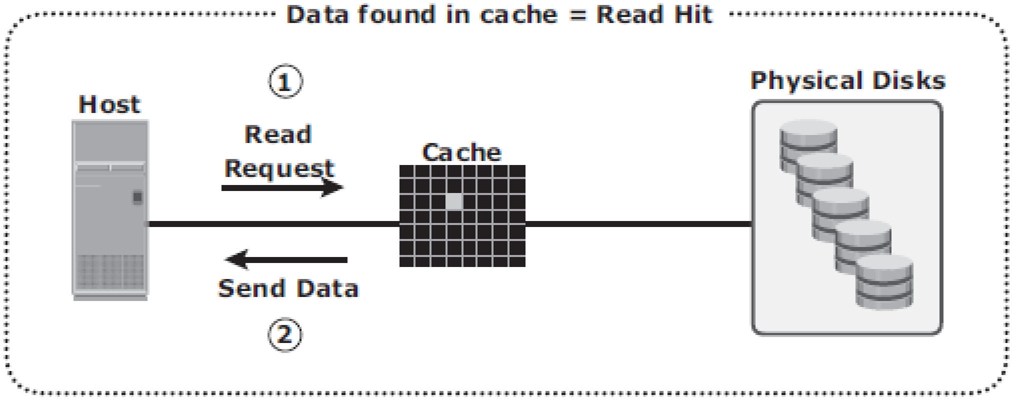
➢ A Pre-fetch, or Read-ahead, algorithm is used when read requests are sequential. In a sequential read request, a contiguous set of associated blocks is retrieved. Several other blocks that have not yet been requested by the host can be read from the disk and placed into cache in advance. When the host subsequently requests these blocks, the read operations will be read hits.

➢ This process significantly improves the response time experienced by the host.

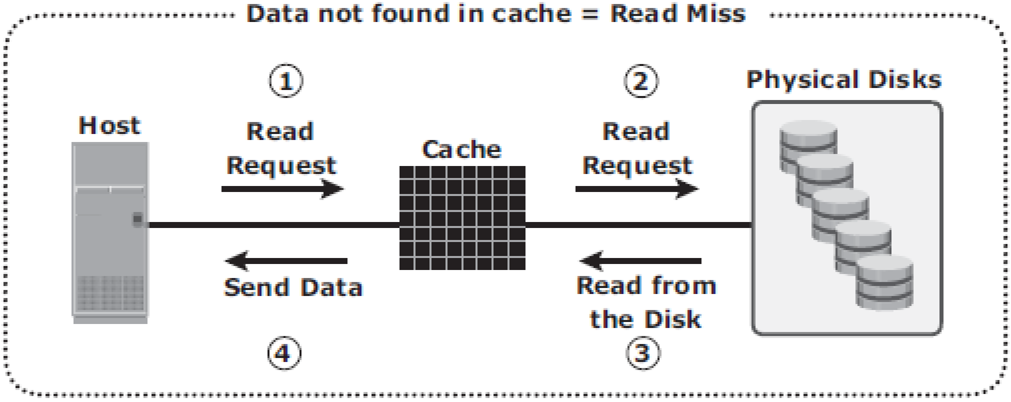
➢ The intelligent storage system offers fixed and variable prefetch sizes.

➢ 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.



(a)



(b)

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.

5 What is a file system? Explain the process of mapping user files to the disk storage.

<u>File System</u>

➢ 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.

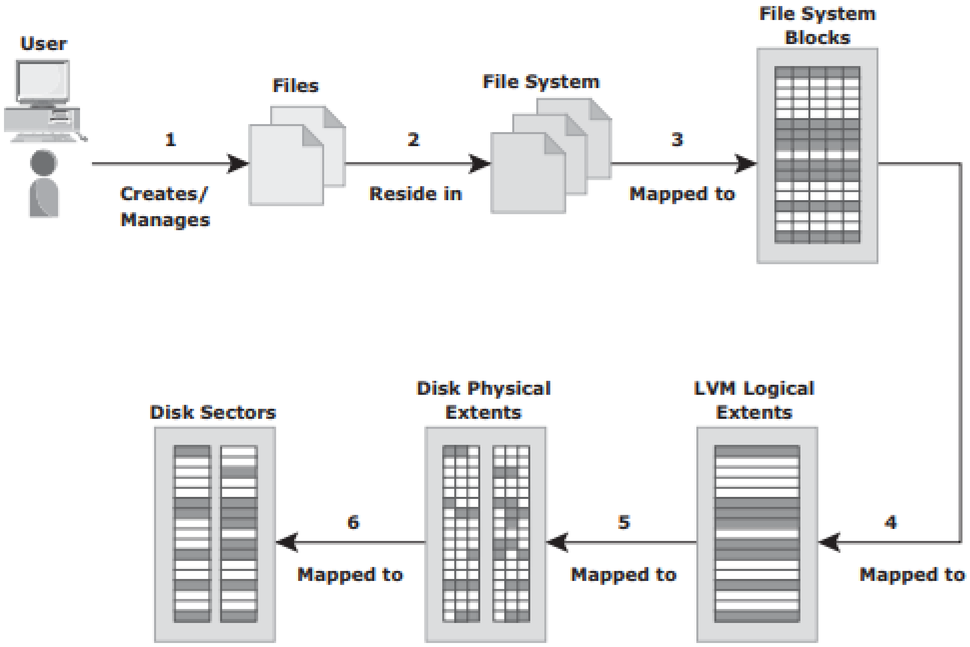
3. The file systems are mapped to file system blocks.

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.



➢ 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 non-journaling 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.

6 What is RAID? Explain the RAID levels with reference to nested RAID, RAID3, RAID5 with neat diagram.

➢ 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 Levels

➢ RAID Level selection is determined by below factors:

✓ Application performance

✓ data availability requirements

✓ cost

➢ RAID Levels are defined on the basis of:

✓ Striping

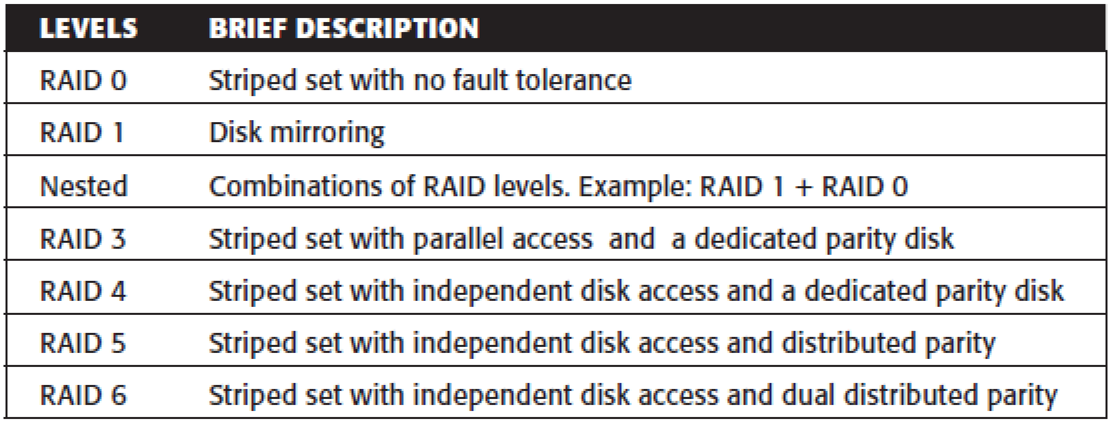
✓ Mirroring

✓ 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



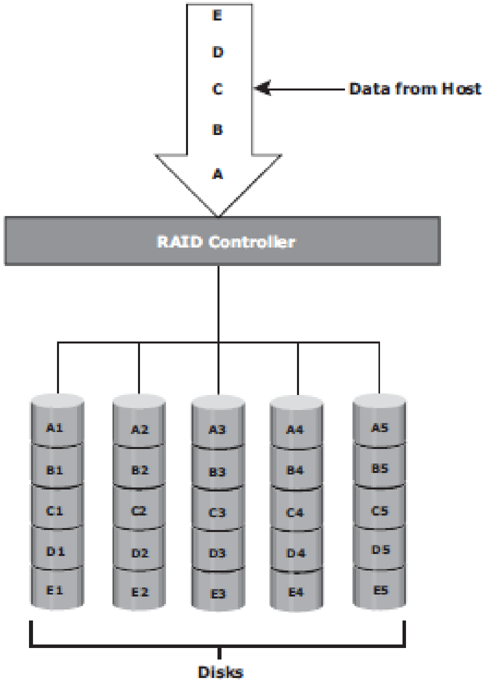
LEVELS	BRIEF DESCRIPTION
RAID 0	Striped set with no fault tolerance
RAID 1	Disk mirroring
Nested	Combinations of RAID levels. Example: RAID 1 + RAID 0
RAID 3	Striped set with parallel access and a dedicated parity disk
RAID 4	Striped set with independent disk access and a dedicated parity disk
RAID 5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

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.



➢ 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.

RAID 1

➢ RAID 1 is based on the mirroring technique.

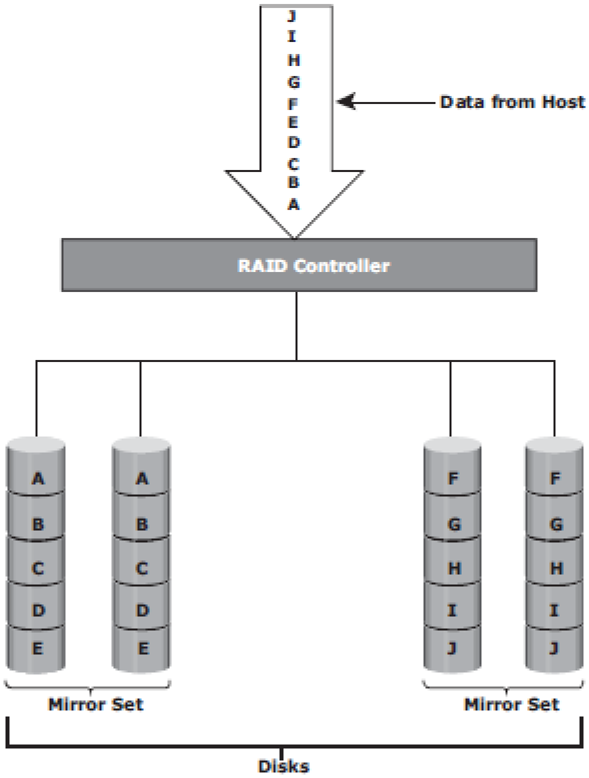
➢ In this RAID configuration, data is mirrored to provide fault tolerance (see Fig 1.15).

➢ 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.



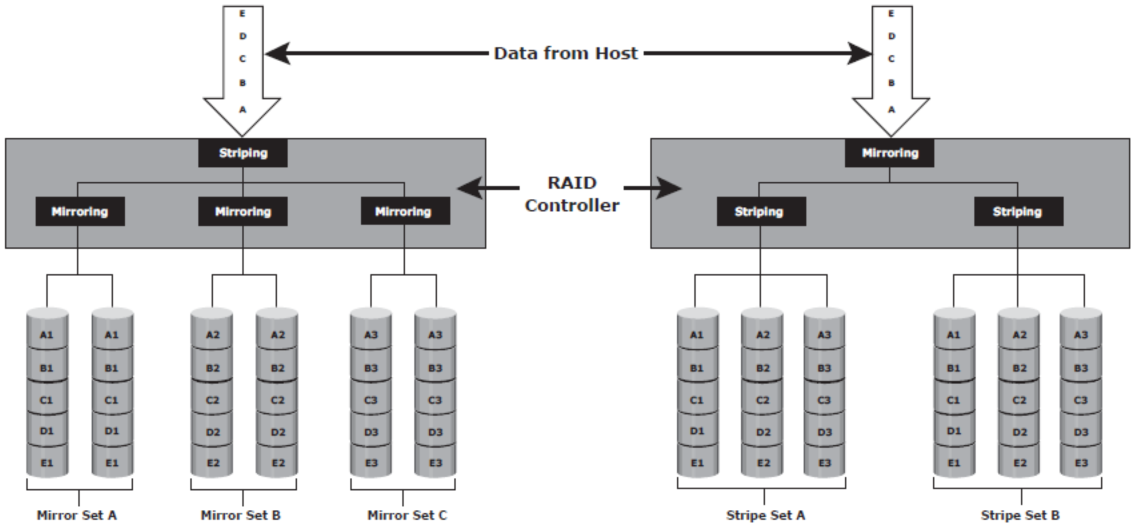
Nested RAID

➢ 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).



(a) RAID 1+0 (b) RAID 0+1

RAID 3

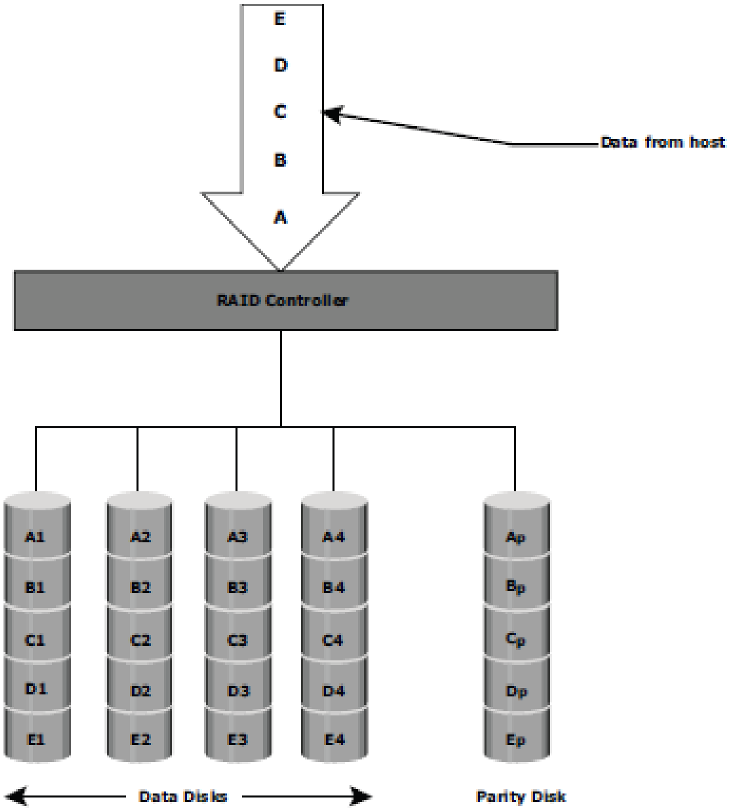
➢ RAID 3 stripes data for high performance and uses parity for improved fault tolerance.

➢ 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



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.

RAID 5

➢ RAID 5 is a versatile RAID implementation.

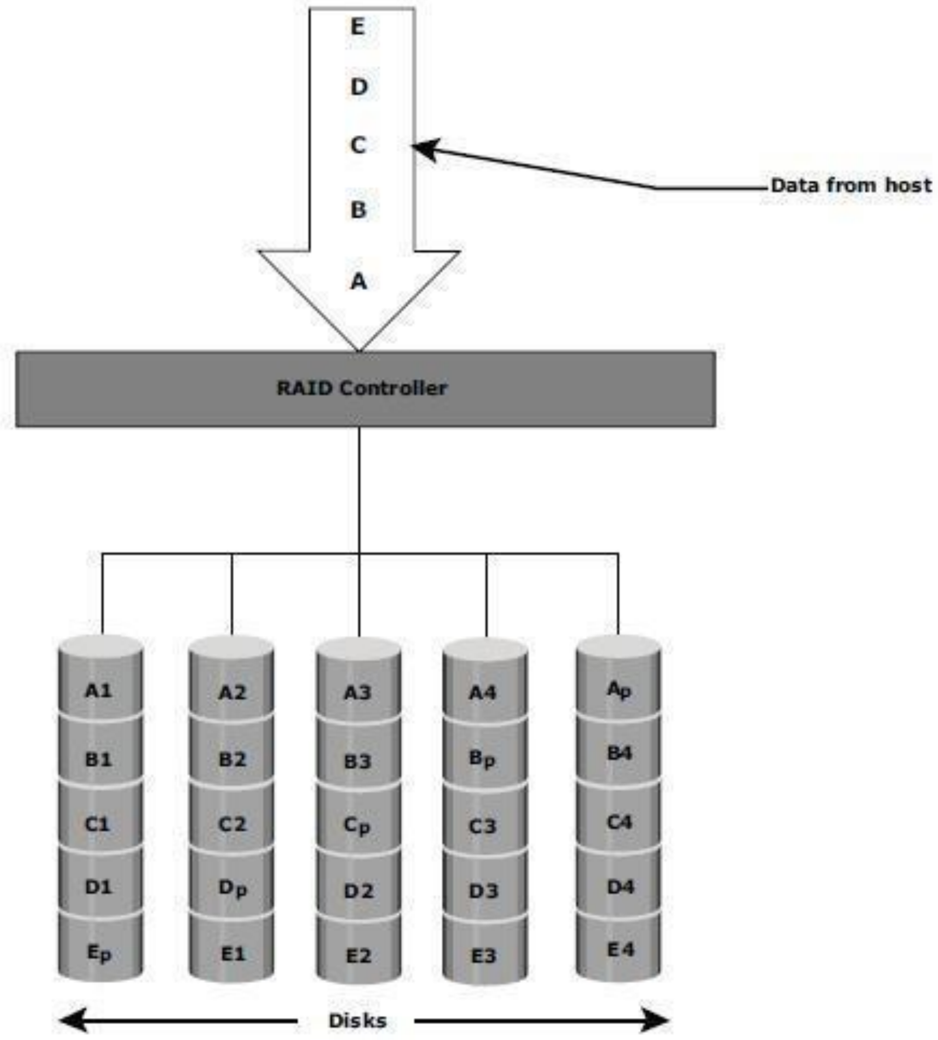
➢ It is similar to RAID 4 because it uses striping. The drives (strips) are also independently accessible.

➢ 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.

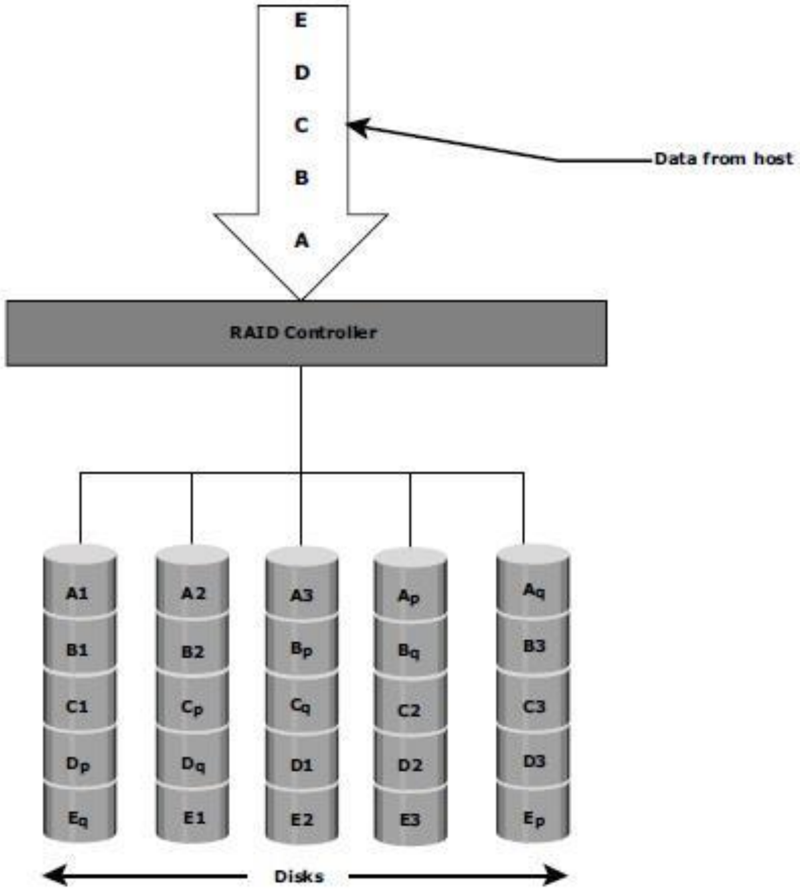


RAID 6

➢ 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



7 Explain with neat diagram the Evolution of storage Architecture.

➢ Historically, organizations had centralized computers (mainframe) and information storage devices (tape reels and disk packs) in their data center.

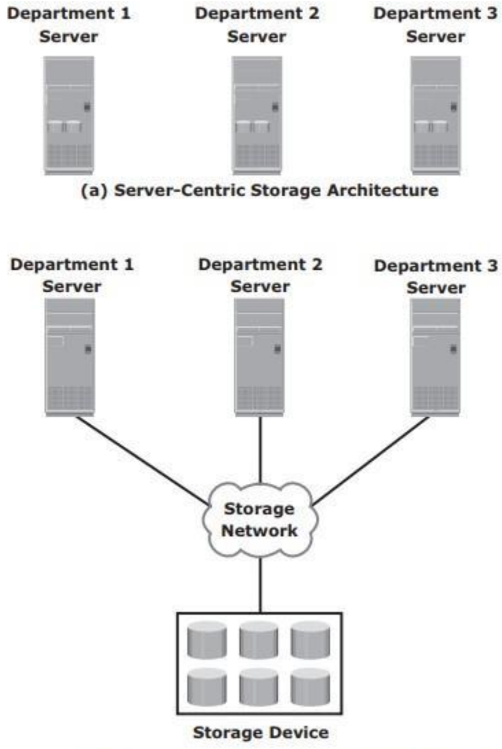
➢ The evolution of open systems and the affordability and ease of deployment that they offer made it possible for business units/departments to have their own servers and storage.

➢ 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]).



(a) Server-Centric Storage Architecture

(b) Information-Centric 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 continue to evolve, which enables organizations to consolidate, protect, optimize, and leverage their data to achieve the highest return on information assets.

8 With diagram explain different RAID Techniques.

➢ There are three RAID techniques

1. striping

2. mirroring

3. parity

Striping

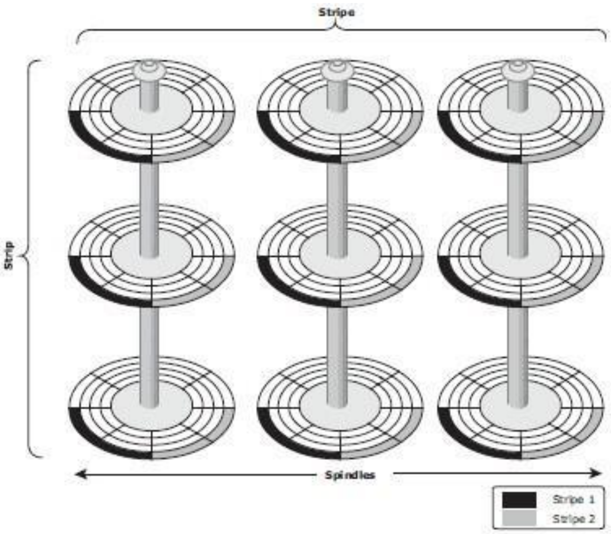
➢ Striping is a technique to spread data across multiple drives (more than one) to use the drives in parallel.

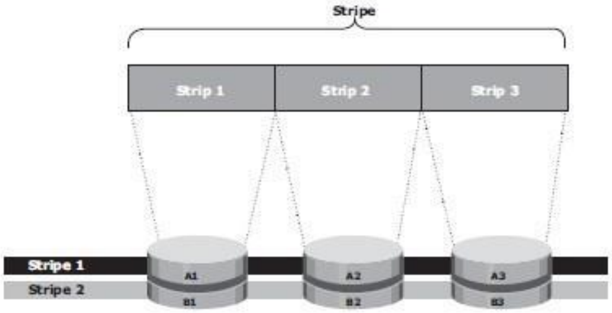
➢ All the read-write heads work simultaneously, allowing more data to be processed in a shorter time and increasing performance, compared to reading and writing from a single disk.

➢ 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.





➢ 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.

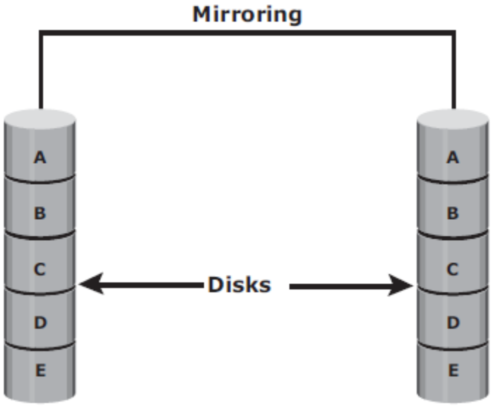
Mirroring

➢ 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.



➢ **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

Parity (Refer from before)