UNIT II

INTELLIGENT STORAGE SYSTEMS AND RAID

INTELLIGENT STORAGE SYSTEM :

Business-critical applications require high levels of performance, availability, security, and scalability. A hard disk drive is a core element of storage that governs the performance of any storage system.

Some of the older disk array technologies could not overcome performance constraints due to the limitations of a hard disk and its mechanical components. RAID technology made an important contribution to enhancing storage performance and reliability, but hard disk drives even with a RAID implementation could not meet performance requirements of today’s applications.

With advancements in technology, a new breed of storage solutions known as an *intelligent storage system* has evolved. These storage systems are configured with large amounts of memory called *cache* and use sophisticated algorithms to meet the I/O requirements of performance- sensitive applications.

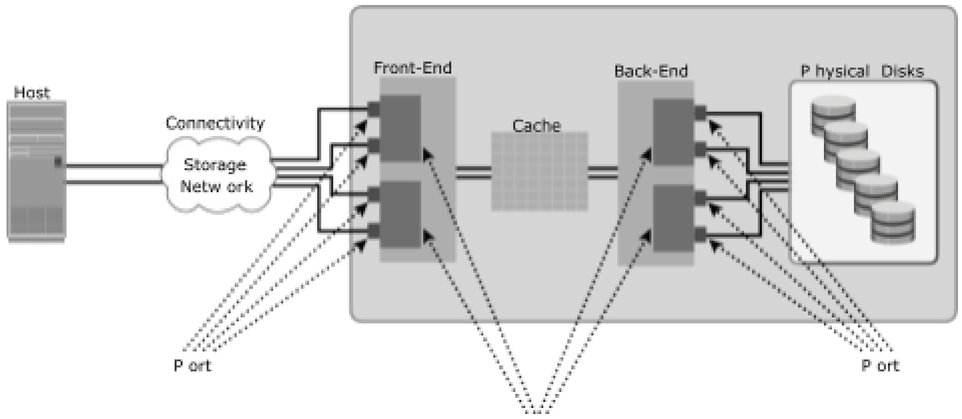
COMPONENTS OF AN INTELLIGENT STORAGE SYSTEM

An intelligent storage system consists of four key components: *front end, cache, back end,* and *physical disks*.

Figure 4-1 illustrates these components and their interconnections.

An I/O request received from the host at the front-end port is processed through cache and the back end, to enable storage and retrieval of data from the physical disk**.** A read request can be serviced directly from cache if the requested data is found in cache.

Intelligent Storage System



Controllers

Figure 4-1: Components of an intelligent storage system

<u>FRONT END</u>

The front end provides the interface between the storage system and the host. It consists of two components: front-end ports and front-end controllers.

The *front-end ports* enable hosts to connect to the intelligent storage system. Each front-end port has processing logic that executes the appropriate transport pro- tocol, such as SCSI, Fibre Channel, or iSCSI, for storage connections.

Redundant ports are provided on the front end for high availability.

Front-end controllers route data to and from cache via the internal data bus.

When cache receives write data, the controller sends an acknowledgment message back to the host. Controllers optimize I/O processing by using command queuing algorithms.

Front-End Command Queuing

• *Command queuing* is a technique implemented on front-end controllers. It determines the execution order of received commands and can reduce unnec- essary drive head movements and improve disk performance.

• When a command is received for execution, the command queuing algorithms assigns a tag that defines a sequence in which commands should be executed.

- With command queuing, multiple commands can be executed concurrently based on the organization of data on the disk, regardless of the order in which the commands were received.

The most commonly used command queuing algorithms are as follows:

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

■ **Seek Time Optimization:** Commands are executed based on optimizing read/write head movements, which may result in reordering of com-mands.

Without seek time optimization, the commands are executed in the order they are received. For example, as shown in Figure 4-2(a), the commands are executed in the order A, B, C and D. The radial movement required by the head to execute C immediately after A is less than what would be required to execute B.

With seek time optimization, the command execution sequence would be A, C, B and D, as shown in Figure 4-2(b).

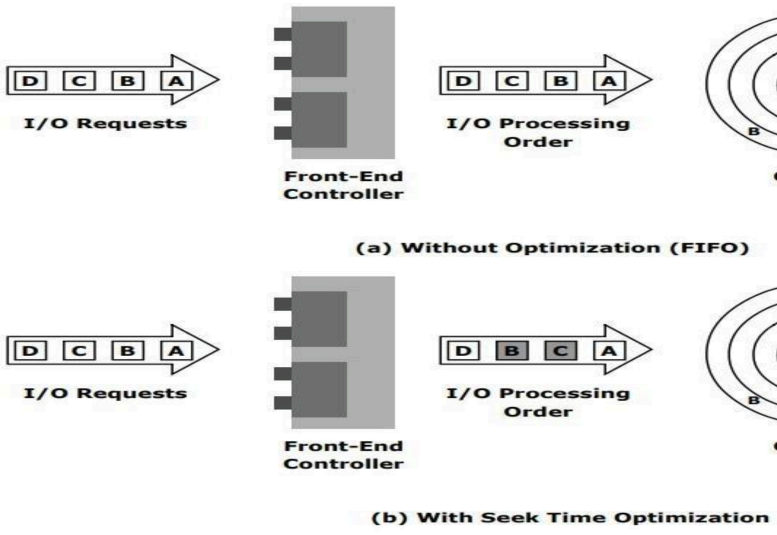


Figure 4-2: Front-end command queuing

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

<u>CACHE</u>

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

Structure of Cache

✓ Cache is organized into pages or slots, which is the smallest unit of cache allo- cation.

✓ The size of a cache page is configured according to the application I/O size. Cache consists of the *data store* and *tag RAM.*

✓ The data store holds the data while tag RAM tracks the location of the data in the data store (see Figure 4-3) and in disk.

✓ Entries in tag RAM indicate where data is found in cache and where the data belongs on the disk. Tag RAM includes a *dirty bit* flag, which indicates whether the data in cache has been committed to the disk or not.

✓ It also contains time-based information, such as the time of last access, which is used to identify cached information that has not been accessed for a long period and may be freed up.

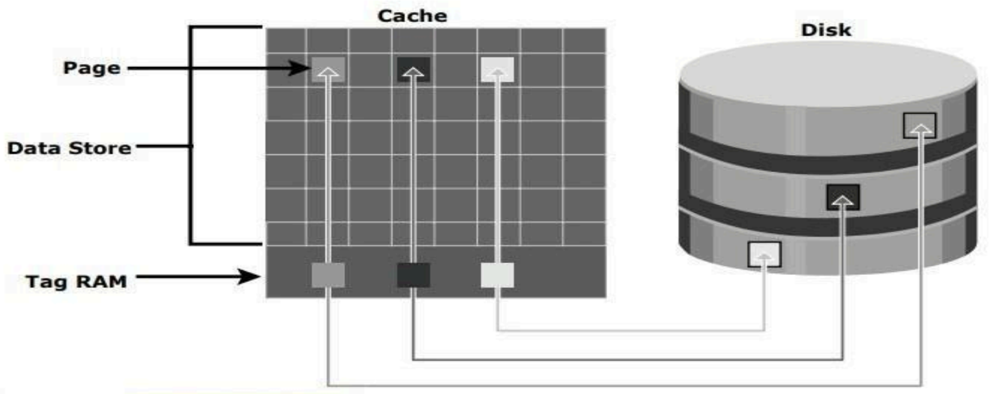


Figure 4-3: Structure of cache

Read Operation with Cache

✓ When a host issues a read request, the front-end controller accesses the tag RAM to determine whether the required data is available in cache.

✓ If the requested data is found in the cache, it is called a *read cache hit* or *read hit* and data is sent directly to the host, without any disk operation (see Figure 4-4[a]). This provides a fast response time to the host (about a millisecond).

✓ If the requested data is not found in cache, it is called a *cache miss* and the data must be read from the disk (see Figure 4-4[b]).

✓ The back-end controller accesses the appropriate disk and retrieves the requested data. Data is then placed in cache and is finally sent to the host through the front- end controller. Cache misses increase I/O response time.

✓ A *pre-fetch,* or *read-ahead,* algorithm is used when read requests are sequential. In a sequential read request, a contiguous set of associated blocks is retrieved. Several other blocks that have not yet been requested by the host can be read from the disk and placed into cache in advance.

✓ The intelligent storage system offers fixed and variable pre-fetch 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.
✓ Read performance is measured in terms of the ***read hit ratio,* or the *hit rate***, usually expressed as a percentage.
This ratio is the number of read hits with respect to the total number of read requests. A higher read hit ratio improves the read performance.

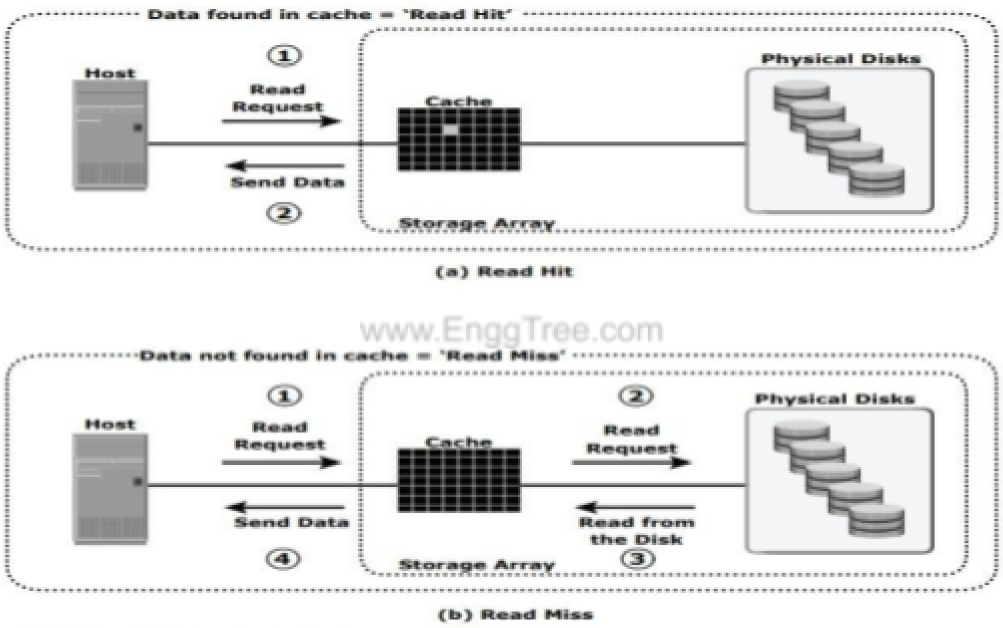


Figure 4-4: Read hit and read miss

Write Operation with Cache

✓ Write operations with cache provide performance advantages over writing directly to disks. When an I/O is written to cache and acknowledged, it is completed in far less time (from the host's perspective) than it would take to write directly to disk.

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

■ **Write-back cache:** Data is placed in cache and an acknowledgment is sent to the host immediately. Later, data from several writes are committed (de-staged) to the disk. Write response times are much faster, as the write operations are isolated from the mechanical delays of the disk. However, uncommitted data is at risk of loss in the event of cache failures.

■ **Write-through cache:** Data is placed in the cache and immediately writ- ten to the disk, and an acknowledgment is sent to the host. Because data is committed to disk as it arrives, the risks of data loss are low but write response time is longer because of the disk operations.

Cache can be bypassed under certain conditions, such as very large size write I/O.

In this implementation, if the size of an I/O request exceeds the pre- defined size, called *write aside size*, writes are sent to the disk directly to reduce the impact of large writes consuming a large cache area.

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 implementa- tion, as only one global set of addresses has to be managed.

Cache Management

✓ Cache is a finite and expensive resource that needs proper management.

✓ Even though intelligent storage systems can be configured with large amounts of cache, when all cache pages are filled, some pages have to be freed up to accom modate new data and avoid performance degradation.

✓ Various cache manage- ment algorithms are implemented in intelligent storage systems :

■ **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.

■ **Most Recently Used (MRU):** An algorithm that is the converse of LRU. In MRU, the pages that have been accessed most recently are freed up or marked for reuse.

✓ As cache fills, the storage system must take action to flush dirty pages (data written into the cahce but not yet written to the disk) in order to manage its availability.

✓ **Flushing** is the process of committing data from cache to the disk. On the basis of the I/O access rate and pattern, high and low levels called *watermarks* are set in cache to manage the flushing process.

✓ *High watermark (HWM)* is the cache utilization level at which the storage system starts high- speed flushing of cache data.

Low watermark (LWM) is the point at which the storage system stops the high-speed or forced flushing and returns to idle flush behavior.

The cache utilization level, as shown in Figure 4-5, drives the mode of flushing to be used:

■ **Idle flushing:** Occurs continuously, at a modest rate, when the cache utilization level is between the high and low watermark.

■ **High watermark flushing:** Activated when cache utilization hits the high watermark. The storage system dedicates some additional resources to flush- ing.

■ **Forced flushing:** Occurs in the event of a large I/O burst when cache reaches 100 percent of its capacity, which significantly affects the I/O response time. In forced flushing, dirty pages are forcibly flushed to disk.

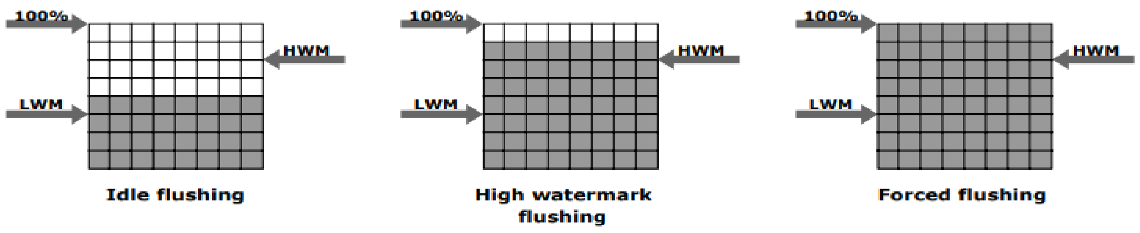


Figure 4-5: Types of flushing

Cache Data Protection

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

This risk of losing uncommitted data held in cache can be mitigated using *cache mirroring* and *cache vaulting*:

■ **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.

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.

▪ **Cache vaulting:** Cache is exposed to the risk of uncommitted data loss due to power failure.

This problem can be addressed in various ways:
power- ing the memory with a battery until AC power is restored or using battery power to write the cache content to the disk.

<u>BACK END</u>

- The *back end* provides an interface between cache and the physical disks. It con- sists of two components: back-end ports and back-end controllers.
- The back end controls data transfers between cache and the physical disks. From cache, data is sent to the back end and then routed to the destination disk. Physical disks are connected to ports on the back end.
- The back end controller communicates with the disks when performing reads and writes

and also provides additional, but lim ited, temporary data storage.

PHYSICAL DISK

A physical disk stores data persistently.
Disks are connected to the back-end with either SCSI or a Fibre Channel interface.
An intelligent storage system enables the use of a mixture of SCSI or Fibre Channel drives and IDE/ATA drives.

Logical Unit Number

✓ Physical drives or groups of RAID protected drives can be logically split into volumes known as logical volumes, commonly referred to as *Logical Unit Numbers* (LUNs).

✓ The use of LUNs improves disk utilization.

✓ For example, without the use of LUNs, a host requiring only 200 GB could be allocated an entire 1TB physical disk. Using LUNs, only the required 200 GB would be allocated to the host, allowing the remaining 800 GB to be allocated to other hosts.

For example, Figure 4-6 shows a RAID set consisting of five disks that have been sliced, or partitioned, into several LUNs. LUNs 0 and 1 are shown in the figure.

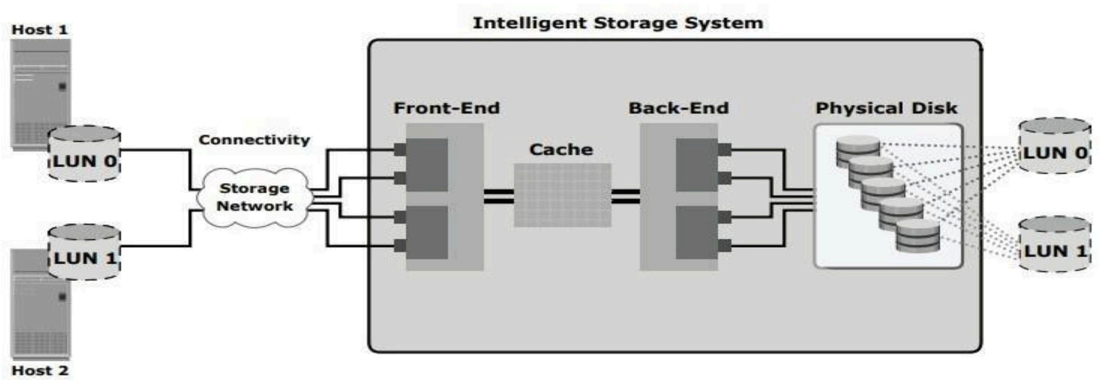


Figure 4-6: Logical unit number

Note how a portion of each LUN resides on each physical disk in the RAID set.

LUNs 0 and 1 are presented to hosts 1 and 2, respectively, as physical vol- umes for storing and retrieving data. Usable capacity of the physical volumes is determined by the RAID type of the RAID set.
The capacity of a LUN can be expanded by aggregating other LUNs with it. The result of this aggregation is a larger capacity LUN, known as a *meta- LUN*. The mapping of LUNs to their physical location on the drives is man- aged by the operating environment of an intelligent storage system.

LUN Masking

✓ *LUN masking* is a process that provides data access control by defining which LUNs a host can access.

✓ LUN masking function is typically implemented at the front end controller. This ensures that volume access by servers is controlled appropriately, preventing unauthorized or accidental use in a distributed environment.

✓ For example, consider a storage array with two LUNs that store data of the sales and finance departments. Without LUN masking, both departments can easily see
and modify each other' data, posing a high risk to data integrity and security.

<u>DISK DRIVE COMPONENTS</u>

• A disk drive uses a rapidly moving arm to read and write data across a flat platter coated with magnetic particles. Data is transferred from the magnetic platter through the R/W head to the computer.
• Several platters are assembled together with the R/W head and controller, most commonly referred to as a *hard disk drive (HDD)*.
○

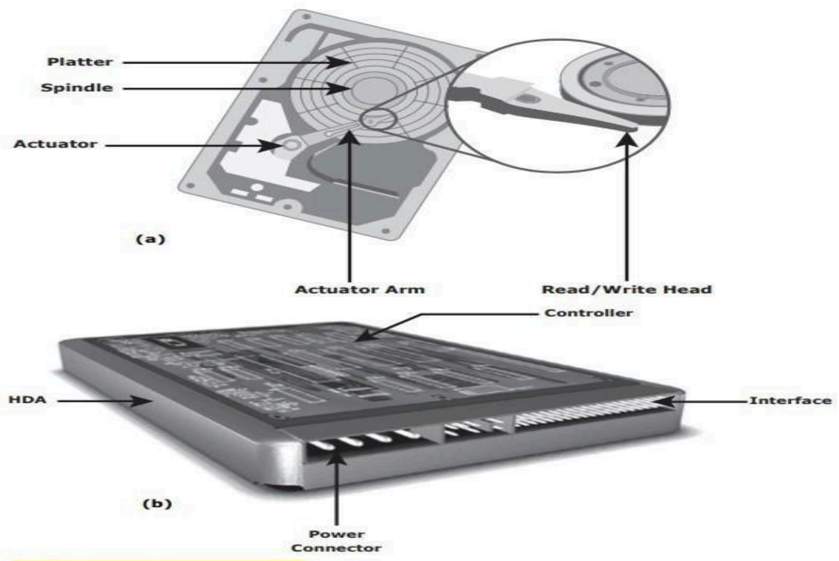


Figure 2-2: Disk Drive Components

- Data can be recorded and erased on a magnetic disk any number of times.

Key components of a disk drive are *platter, spindle, read/write head, actuator arm assembly, and controller* (Figure 2-2):

PLATTER

✓ A typical HDD consists of one or more flat circular disks called *platters* (Figure 2-3). The data is recorded on these platters in binary codes (0s and 1s).

✓ The set of rotating platters is sealed in a case, called a *Head Disk Assembly (HDA)*. A platter is a rigid, round disk coated with magnetic material on both surfaces (top and bottom).

✓ The data is encoded by polarizing the magnetic area, or domains, of the disk surface. Data can be written to or read from both surfaces of the platter.

✓ The number of platters and the storage capacity of each platter determine the total capacity of the drive.

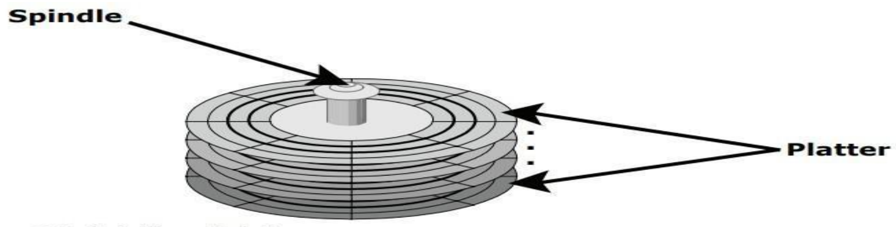


Figure 2-3: Spindle and platter

SPINDLE

✓ A spindle connects all the platters, as shown in Figure 2-3, and is connected to a motor. The motor of the spindle rotates with a constant speed.

✓ The disk platter spins at a speed of several thousands of revolutions per minute (rpm). Disk drives have spindle speeds of 7,200 rpm, 10,000 rpm, or 15,000 rpm. Disks used on current storage systems have a platter diameter of 3.5" (90 mm).

✓ When the platter spins at 15,000 rpm, the outer edge is moving at around 25 percent of the speed of sound.

READ/WRITE HEAD

✓ *Read/Write (R/W) heads*, shown in Figure 2-4, read and write data from or to a platter.

✓ Drives have two R/W heads per platter, one for each surface of the platter.

✓ The R/W head changes the magnetic polarization on the surface ofthe platter when writing data. While reading data, this head detects magnetic polarization on the surface of the platter.

✓ During reads and writes, the R/W head senses the magnetic polarization and never touches the surface of the platter. When the spindle is rotating, there is a microscopic air gap between the R/W heads and the platters, known as the *head flying height*.

✓ This air gap is removed when the spindle stops rotating and the R/W head rests on a special area on the platter near the spindle. This area is called the *landing zone*. The landing zone is coated with a lubricant to reduce friction between the head and the platter.

✓ The logic on the disk drive ensures that heads are moved to the landing zone before they touch the surface. If the drive malfunctions and the R/W head accidentally touches the surface of the platter outside the landing zone, a *head crash* occurs.

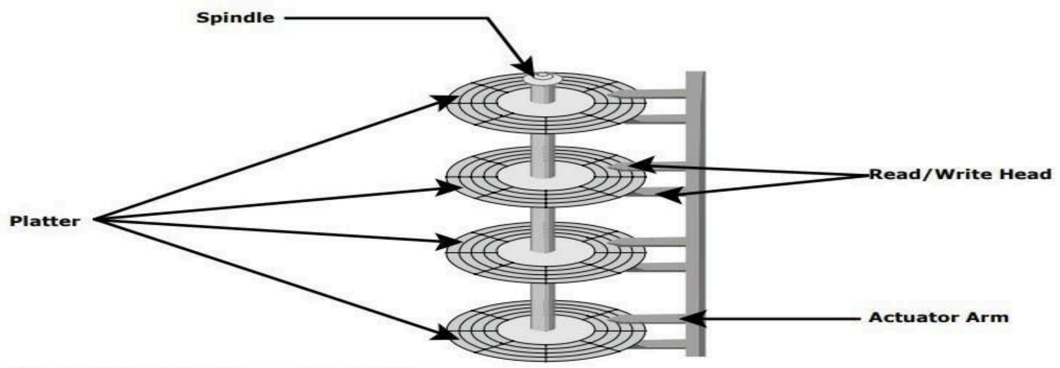


Figure 2-4: Actuator arm assembly

ACTUATOR ARM ASSEMBLY

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

CONTROLLER

✓ The *controller* (see Figure 2-2 [b]) is a printed circuit board, mounted at the bottom of a disk drive. It consists of a microprocessor, internal memory, circuitry, and firmware.

✓ The firmware controls power to the spindle motor and the speed of the motor. It also manages communication between the drive and the host.

✓ In addition, it controls the R/W operations by moving the actuator arm and switching between different R/W heads, and performs the optimization of data access.

Physical Disk Structure

• Data on the disk is recorded on *tracks*, which are concentric rings on the platter around the spindle, as shown in Figure 2-5. The tracks are numbered, starting from zero, from the outer edge of the platter. The number of *tracks per inch (TPI)* on the platter (or the *track density*) measures how tightly the tracks are packed on a platter.

• Each track is divided into smaller units called *sectors*. A sector is the smallest, individually addressable unit of storage. The track and sector structure is written on the platter by the drive manufacturer using a formatting operation. The number of sectors per track varies according to the specific drive.

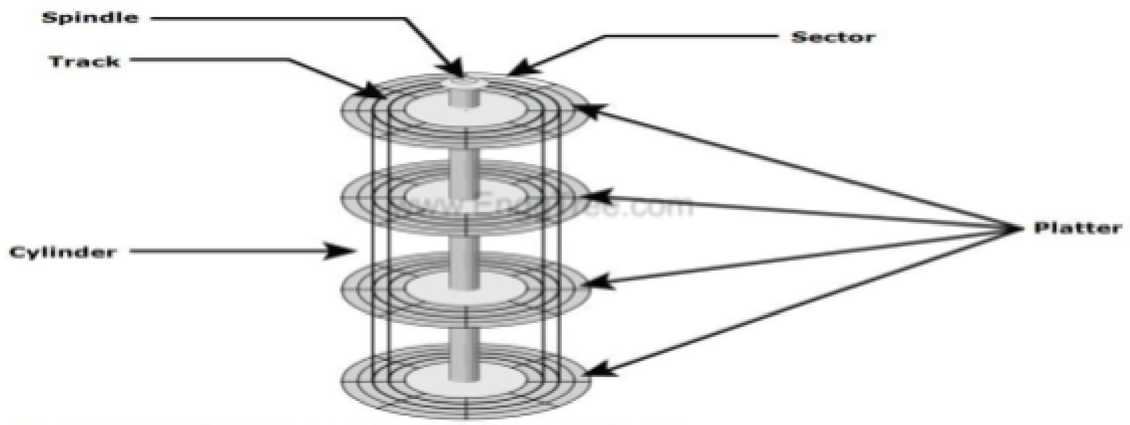


Figure 2-5: Disk structure: sectors, tracks, and cylinders

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

Consequently, there is a difference between the capacity of an unformatted disk and a format- ted one. Drive manufacturers generally advertise the unformatted capacity — for example, a disk advertised as being 500GB will only hold 465.7GB of user data, and the remaining 34.3GB is used for *metadata*.
A cylinder is the set of identical tracks on both surfaces of each drive plat- ter.
The location of drive heads is referred to by cylinder number, not by track number.

Zoned Bit Recording

- Because the platters are made of concentric tracks, the outer tracks can hold more data than the inner tracks, because the outer tracks are physically longer than the inner tracks, as shown in Figure 2-6 (a).
- On older disk drives, the outer tracks had the same number of sectors as the inner tracks, so data density was low on the outer tracks. This was an inefficient use of available space.

Zone bit recording utilizes the disk efficiently.
As shown in Figure 2-6 (b), this mechanism groups tracks into zones based on their distance from the center of the disk. The zones are numbered, with the outermost zone being zone 0.

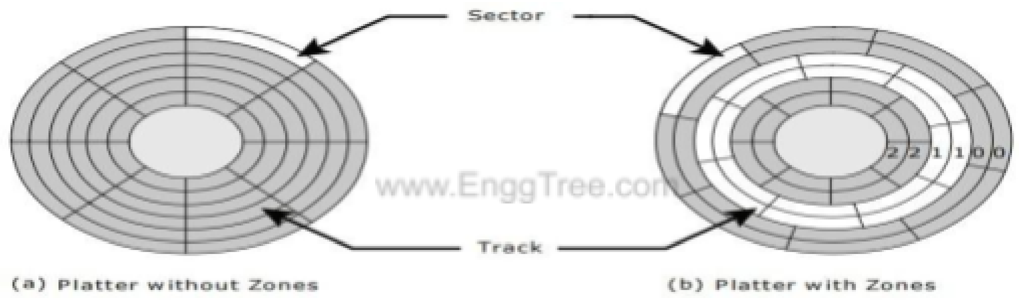


Figure 2-6: Zoned bit recording

➔ LOGICAL BLOCK ADDRESSING

Earlier drives used physical addresses consisting of the *cylinder, head, and sector (CHS)* number to refer to specific locations on the disk, as shown in Figure 2-7 (a), and the host operating system had to be aware of the geometry of each disk being used. *Logical block addressing (LBA)*, shown in Figure 2-7 (b), simplifies addressing by using a linear address to access physical blocks of data.

The disk controller translates LBA to a CHS address, and the host only needs to know the size of the disk drive in terms of the number of blocks. The logical blocks are mapped to physical sectors on a 1:1 basis.

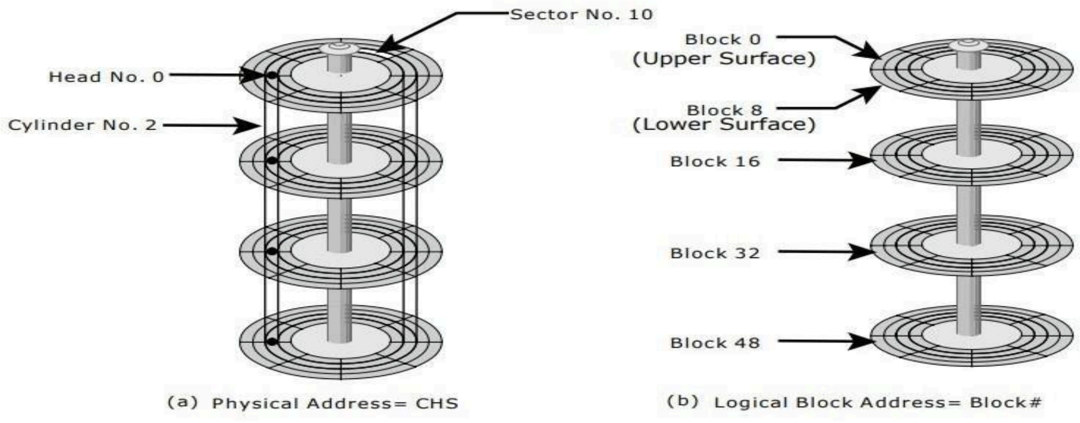


Figure 2-7: Physical address and logical block address

In Figure 2-7 (b), the drive shows eight sectors per track, eight heads, and four cylinders. This means a total of 8 × 8 × 4 = 256 blocks, so the block number ranges from 0 to 255. Each block has its own unique address. Assuming that the sector holds 512 bytes, a 500 GB drive with a formatted capacity of 465.7 GB will have in excess of 976,000,000 blocks.

➔ DISK DRIVE PERFORMANCE

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

Disk Service Time

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

Seek Time

The *seek time* (also called *access time*) describes the time taken to position the R/W heads across the platter with a radial movement (moving along the radius of the platter).

In other words, it is the time taken to reposition and settle the arm and the head over the correct track. The lower the seek time, the faster the I/O operation.

Disk vendors publish the following seek time specifications:

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

Each of these specifications is measured in milliseconds. The average seek time on a modern disk is typically in the range of 3 to 15 milliseconds. Seek time has more impact on the read operation of random tracks rather than adjacent tracks.

To minimize the seek time, data can be written to only a subset of the available cylinders. This results in lower usable capacity than the actual capacity of the drive. For example, a 500 GB disk drive is set up to use only the first 40 percent of the cylinders and is effectively treated as a 200 GB drive. This is known as *short-stroking* the drive.

Rotational Latency

To access data, the actuator arm moves the R/W head over the platter to a par- ticular track while the platter spins to position the requested sector under the R/W head**.** The time taken by the platter to rotate and position the data under the R/W head is called *rotational latency*.
This latency depends on the rotation speed of the spindle and is measured in milliseconds. The average rotational latency is one-half of the time taken for a full rotation.
Average rotational latency is around 5.5 ms for a 5,400-rpm drive, and around

2.0 ms for a 15,000-rpm drive.

Data Transfer Rate

The *data transfer rate* (also called *transfer rate*) refers to the average amount of data per unit time that the drive can deliver to the HBA.

In a *read operation*, the data first moves from disk platters to R/W heads, and then it moves to the drive's internal *buffer*. Finally, data moves from the buffer through the interface to the host HBA.

In a *write operation*, the data moves from the HBA to the internal buffer of the disk drive through the drive's interface. The data then moves from the buffer to the R/W heads. Finally, it moves from the R/W heads to the platters.

The data transfer rates during the R/W operations are measured in terms of internal and external transfer rates, as shown in Figure 2-8.

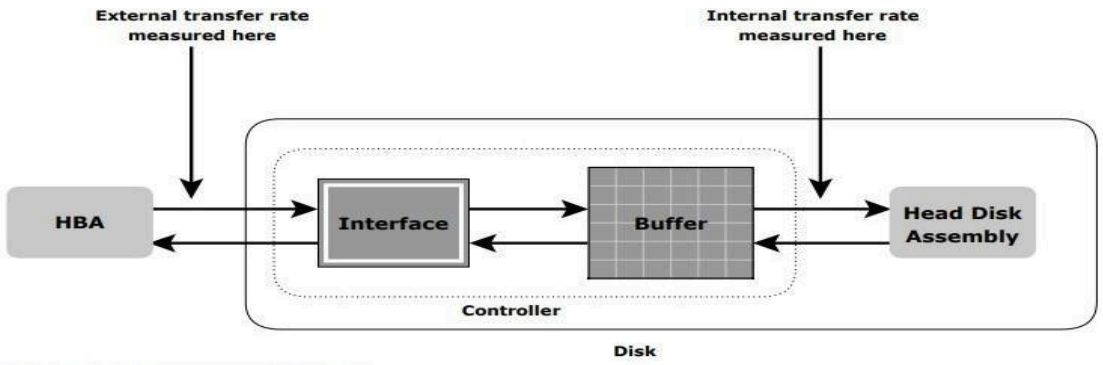

External transfer rate measured here
Internal transfer rate measured here
HBA
Interface
Buffer
Head Disk Assembly
Controller
Disk

Figure 2-8: Data transfer rate

Internal transfer rate is the speed at which data moves from a single track of a platter's surface to internal buffer (cache) of the disk. Internal transfer rate takes into account factors such as the seek time.

External transfer rate is the rate at which data can be moved through the interface to the HBA. External transfer rate is generally the advertised speed of the interface, such as 133 MB/s for ATA. The sustained external transfer rate is lower than the interface speed.

<u>RAID</u> (Redundant array of Independent Disks)

RAID is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure.

There are two types of RAID implementation, hardware and software.

<u>Software RAID</u>

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

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

■ **Supported features:** Software RAID does not support all RAID levels.

■ **Operating system compatibility:** Software RAID is tied to the host operat- ing 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.

<u>Hardware RAID</u>

In *hardware RAID* implementations, a specialized hardware controller is imple- mented either on the host or on the array. These implementations vary in the way the storage array interacts with the host.
Controller card RAID is host-based hardware RAID implementation in which a specialized RAID controller is installed in the host and HDDs are

connected to it.

The RAID Controller interacts with the hard disks using a PCI bus. Manufacturers also integrate RAID controllers on motherboards. This integra- tion reduces the overall cost of the system, but does not provide the flexibility required for high-end storage systems.

The external RAID controller is an array-based hardware RAID. It acts as an interface between the host and disks. It presents storage volumes to the host, which manage the drives using the supported protocol. Key functions of RAID controllers are:

- Management and control of disk aggregations
- Translation of I/O requests between logical disks and physical disks
- Data regeneration in the event of disk failures

RAID Array Components

RAID array is an enclosure that contains a number of HDDs and the supporting hardware and software to implement RAID. HDDs inside a RAID array are usually contained in smaller sub- enclosures.

These sub-enclosures, or *physical arrays*, hold a fixed number of HDDs, and may also include other supporting hardware, such as power supplies. 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* (see Figure 3-1).

Logical arrays are comprised of logical volumes (LV). The operating system recognizes the LVs as if they are physical HDDs managed by the RAID controller.

The number of HDDs in a logical array depends on the RAID level used. Configurations could have a logical array with multiple physical arrays or a physical array with multiple logical arrays.

RAID LEVELS

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

Striping

A RAID set is a group of disks. Within each disk, a predefined number of contiguously addressable disk blocks are defined as *strips*. The set of aligned strips that spans across all the disks within the RAID set is called a *stripe*. Figure 3-2 shows physical and logical representations of a striped RAID set.
Strip size (also called *stripe depth*) describes the number of blocks in a *strip*, and is the maximum amount of data that can be written to or read from a single HDD in the set before the next HDD is accessed, assuming that the accessed data starts at the beginning of the strip.
Note that all strips in a stripe have the same number of blocks, and decreasing strip size means that data is broken into smaller pieces when spread across the disks.

Stripe size is a multiple of strip size by the number of HDDs in the RAID set.
Stripe width refers to the number of data strips in a stripe.

Striped RAID does not protect data unless parity or mirroring is used. However, striping may significantly improve I/O performance. Depending on the type of RAID implementation, the RAID controller can be configured to access data across multiple HDDs simultaneously.

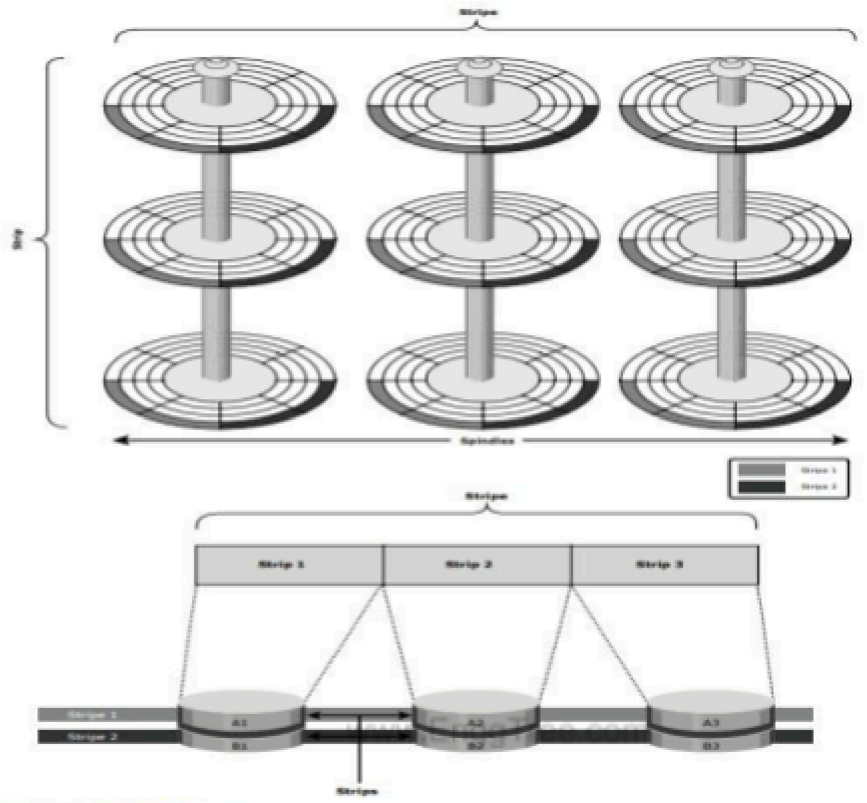


Figure 3-2: Striped RAID set

Mirroring

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

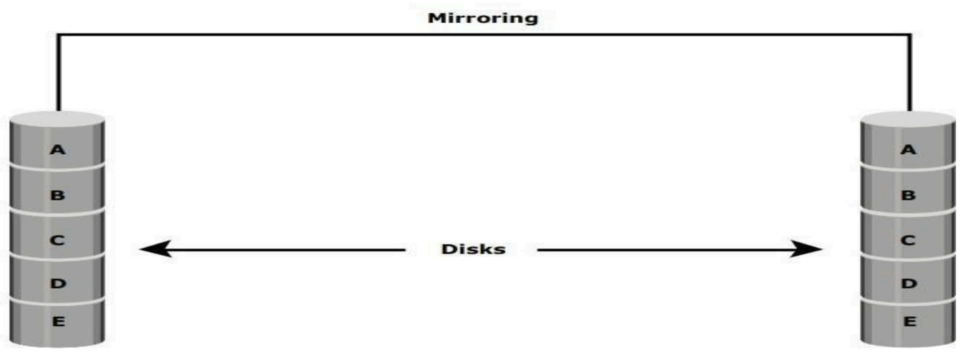


Figure 3-3: Mirrored disks in an array

When the failed disk is replaced with a new disk, the controller copies the data from the surviving disk of the mirrored pair. This activity is transparent to the host.
In addition to providing complete data redundancy, mirroring enables faster recovery

from disk failure. However, disk mirroring provides only data protection and is not a substitute for data backup. Mirroring constantly captures changes in the data, whereas a backup captures point- in-time images of data.

Mirroring involves duplication of data — the amount of storage capacity needed is twice the amount of data being stored. Therefore, mirroring is con- sidered expensive and is preferred for mission-critical applications that cannot afford data loss.

Mirroring improves read performance because read requests can be serviced by both disks. However, write performance deteriorates, as each write request manifests as two writes on the HDDs. In other words, mirroring does not deliver the same levels of write performance as a striped RAID.

<u>Parity</u>

Parity is a method of protecting striped data from HDD failure without the cost of mirroring. An additional HDD is added to the stripe width to hold parity, a mathematical construct that allows re-creation of the missing data.

Parity is a redundancy check that ensures full protection of data without maintaining a full set of duplicate data.
Parity information can be stored on separate, dedicated HDDs or distributed across all the drives in a RAID set. Figure 3-4 shows a parity RAID. The first four disks, labeled *D,* contain the data.
The fifth disk, labeled *P*, stores the parity information, which in this case is the sum of the elements in each row. Now, if one of the *D*s fails, the missing value can be calculated by subtracting the sum of the rest of the elements from the parity value.

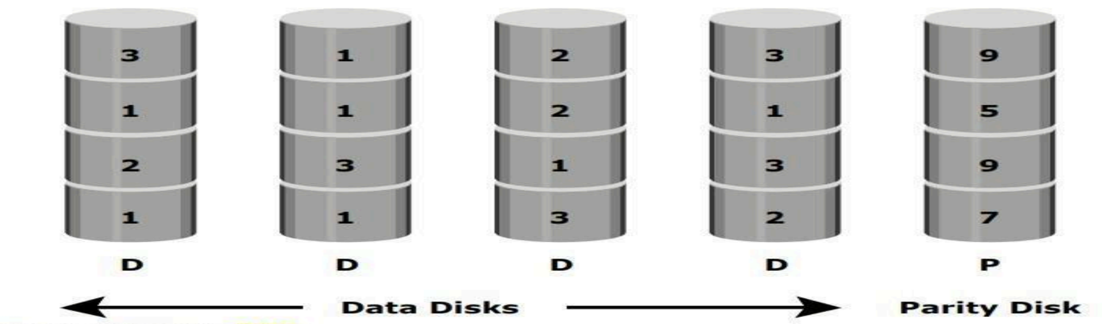


Figure 3-4: Parity RAID

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

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

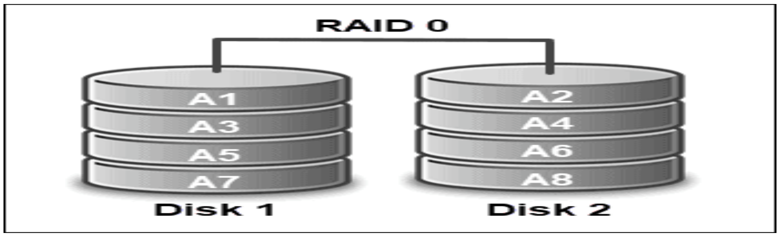
Table 3-1: Raid Levels

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

RAID 0: Striping

RAID 0, also known as a striped set or a striped volume, requires a minimum of two disks. The disks are merged into a single large volume where data is stored evenly across the number of disks in the array.
This process is called disk striping and involves splitting data into blocks and writing it simultaneously/sequentially on multiple disks. Configuring

the striped disks as a single partition increases performance since multiple disks do reading and writing operations simultaneously.
Therefore, RAID 0 is generally implemented to improve speed and efficiency.



It is important to note that if an array consists of disks of different sizes, each will be limited to the smallest disk size in the setup. This means that an array composed of two disks, where one is 320 GB, and the other is 120 GB, actually has the capacity of 2 x 120 GB (or 240 GB in total).

Certain implementations allow you to utilize the remaining 200 GB for different use.
Additionally, developers can implement multiple controllers (or even one per disk) to improve performance.
RAID 0 is the most affordable type of redundant disk configuration and is relatively easy to set up. Still, it does not include any redundancy, fault tolerance, or party in its composition.
Hence, problems on any of the disks in the array can result in complete data loss. This is why it should only be used for non-critical storage, such as temporary files backed up somewhere else.

Advantages of RAID 0
- Cost-efficient and straightforward to implement.
- Increased read and write performance.
- No overhead (total capacity use).

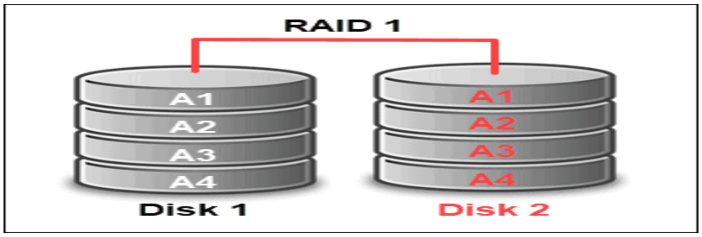
Disadvantages of RAID 0

- Doesn't provide fault tolerance or redundancy.

When Raid 0 Should Be Used
RAID 0 is used when performance is a priority and reliability is not. If you want to utilize your drives to the fullest and don't mind losing data, opt for RAID 0.
On the other hand, such a configuration does not necessarily have to be unreliable. You can set up disk striping on your system along with another RAID array that ensures data protection and redundancy.

RAID 1: Mirroring

RAID 1 is an array consisting of at least two disks where the same data is stored on each to ensure redundancy. The most common use of RAID 1 is setting up a mirrored pair consisting of two disks in which the contents of the first disk is mirrored in the second. This is why such a configuration is also called mirroring.
Unlike with RAID 0, where the focus is solely on speed and performance, the primary goal of RAID 1 is to provide redundancy. It eliminates the possibility of data loss and downtime by replacing a failed drive with its replica.



In such a setup, the array volume is as big as the smallest disk and operates as long as one drive is operational. Apart from reliability, mirroring enhances read performance as a request can be handled by any of the drives in the array. On the other hand, the write performance remains the same as with one disk and is equal to the slowest disk in the
configuration.

Advantages of RAID 1

- Increased read performance.
- Provides redundancy and fault tolerance.
- Simple to configure and easy to use.

Disadvantages of RAID 1

- Uses only half of the storage capacity.
- More expensive (needs twice as many drivers).
- Requires powering down your computer to replace failed drive.

When Raid 1 Should Be Used

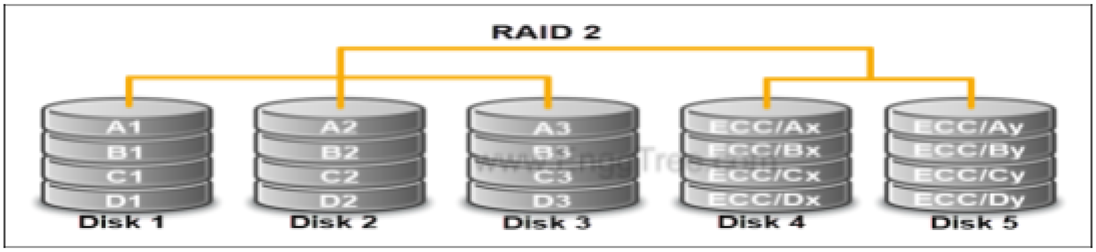
RAID 1 is used for mission-critical storage that requires a minimal risk of data loss. Accounting systems often opt for RAID 1 as they deal with critical data and require high reliability.

It is also suitable for smaller servers with only two disks, as well as if you are searching for a simple configuration you can easily set up (even at home).

Raid 2: Bit-Level Striping with Dedicated Hamming-Code Parity

RAID 2 is rarely used in practice today. It combines bit-level striping with error checking and information correction. This RAID implementation requires two groups of disks – one for writing the data and another for writing error correction codes. RAID 2 also requires a special controller for the synchronized spinning of all disks.

Instead of data blocks, RAID 2 stripes data at the bit level across multiple disks. Additionally, it uses the Humming error ode correction (ECC) and stores this information on the redundancy disk.



The array calculates the error code correction on the fly. While writing the data, it strips it to the data disk and writes the code to the redundancy disk. On the other hand, while reading data from the disk, it also reads from the redundancy disk to verify the data and make corrections if needed.

Advantages of RAID 2

- Reliability.
- The ability to correct stored information.

Disadvantages of RAID 2

- Expensive.
- Difficult to implement.
- Require entire disks for ECC.

When Raid 2 Should Be Used

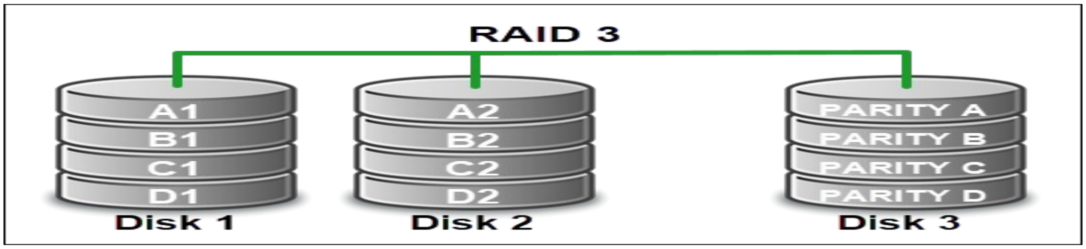
RAID 2 is not a common practice today as most of its features are now available on modern hard disks. Due to its cost and implementation requirements, this RAID level never became popular among developers.

Raid 3: Bit-Level Striping with Dedicated Parity

Like RAID 2, RAID 3 is rarely used in practice. This RAID implementation utilizes bit-level striping and a dedicated parity disk. Because of this, it requires at least three drives, where two are used for storing data strips, and one is used for parity.

To allow synchronized spinning, RAID 3 also needs a special controller.

Due to its configuration and synchronized disk spinning, it achieves better performance rates with sequential operations than random read/write operations.



Advantages of RAID 3
- Good throughput when transferring large amounts of data.
- High efficiency with sequential operations.
- Disk failure resiliency.

Disadvantages of RAID 3
- Not suitable for transferring small files.
- Complex to implement.
- Difficult to set up as software RAID.

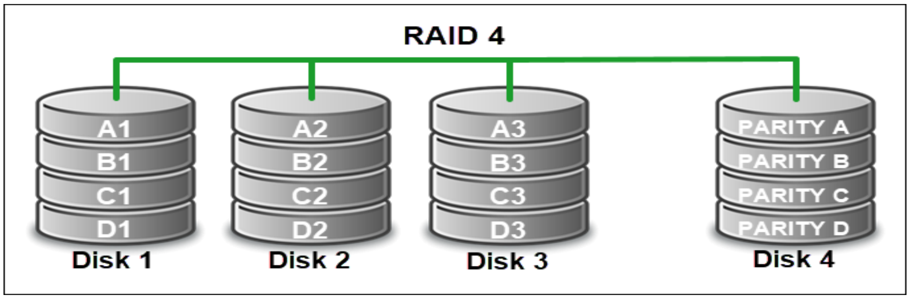
When Raid 3 Should Be Used

RAID 3 is not commonly used today. Its features are beneficial to a limited number of use cases requiring high transfer rates for long sequential reads and writes (such as video editing and production).

Raid 4: Block-Level Striping with Dedicated Parity

RAID 4 is another unpopular standard RAID level. It consists of block-level data striping across two or more independent diss and a dedicated parity disk.

The implementation requires at least three disks – two for storing data strips and one dedicated for storing parity and providing redundancy. As each disk is independent and there is no synchronized spinning, there is no need for a controller.



RAID 4 configuration is prone to bottlenecks when storing parity bits for each data block on a single drive. Such system bottlenecks have a large impact on system performance.

Advantages of RAID 4

- Fast read operations.
- Low storage overhead.
- Simultaneous I/O requests. Disadvantages of RAID 4
- Bottlenecks that have big effect on overall performance.
- Slow write operations.
- Redundancy is lost if the parity disk fails.

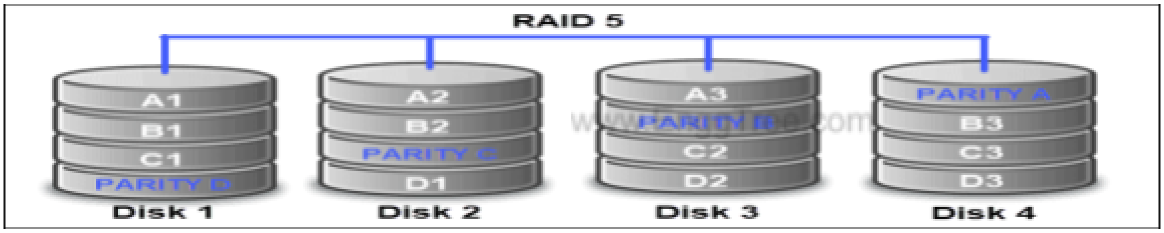
When Raid 4 Should Be Used

Considering its configuration, RAID 4 works best with use cases requiring sequential reading and writing data processes of huge files. Still, just like with RAID 3, in most solutions, RAID 4 has been replaced with RAID 5.

Raid 5: Striping with Parity

RAID 5 is considered the most secure and most common RAID implementation. It combines striping and parity to provide a fast and reliable setup. Such a configuration gives the user storage usability as with RAID 1 and the performance efficiency of RAID 0.

This RAID level consists of at least three hard drives (and at most, 16). Data is divided into data strips and distributed across different disks in the array. This allows for high performance rates due to fast read data transactions which can be done simultaneously by different drives in the array.



Parity bits are distributed evenly on all disks after each sequence of data has been saved. This feature ensures that you still have access to the data from parity bits in case of a failed drive. Therefore, RAID 5 provides redundancy through parity bits instead of mirroring.

Advantages of RAID 5

- High performance and capacity.
- Fast and reliable read speed.
- Tolerates single drive failure.

Disadvantages of RAID 5

- Longer rebuild time.
- Uses half of the storage capacity (due to parity).
- If more than one disk fails, data is lost.
- More complex to implement.

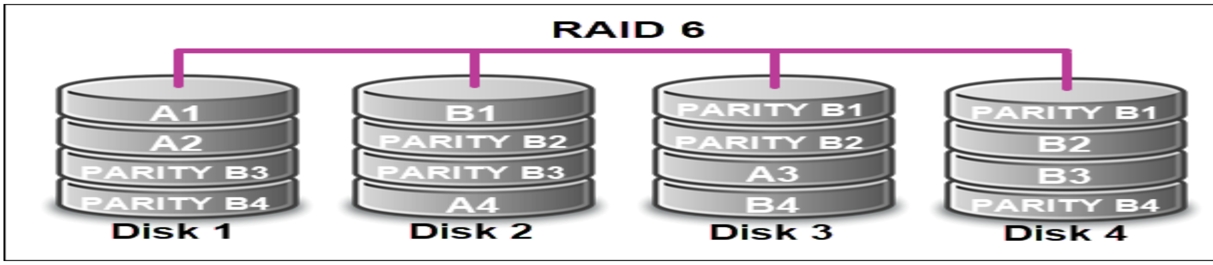
When Raid 5 Should Be Used

RAID 5 is often used for file and application servers because of its high efficiency and optimized storage. Additionally, it is the best, cost-effective solution if continuous data access is a priority and/or you require installing an operating system on the array.

Raid 6: Striping with Double Parity

RAID 6 is an array similar to RAID 5 with an addition of its double parity feature. For this reason, it is also referred to as the double-parity RAID.

This setup requires a minimum of four drives. The setup resembles RAID 5 but includes two additional parity blocks distributed across the disk. Therefore, it uses block-level striping to distribute the data across the array and stores two parity blocks for each data block.



Block-level striping with two parity blocks allows two disk failures before any data is lost. This means that in an event where two disks fail, RAID can still reconstruct the required data.

Its performance depends on how the array is implemented, as well as the total number of drives.
Write operations are slower compared to other configurations due to its double parity feature.

Advantages of RAID 6
- High fault and drive-failure tolerance.
- Storage efficiency (when more than four drives are used).
- Fast read operations. Disadvantages of RAID 6
- Rebuild time can take up to 24 hours.
- Slow write performance.
- Complex to implement.
- More expensive.

When Raid 6 Should Be Used
RAID 6 is a good solution for mission-critical applications where data loss cannot be tolerated.
Therefore, it is often used for data management in defense sectors,

healthcare, and banking.

Raid 10: *Mirroring with Striping*

RAID 10 is part of a group called nested or hybrid RAID, which means it is a combination of two different RAID levels. In the case of RAID 10, the array combines level 1 mirroring and level 0 striping. This RAID array is also known as RAID 1+0.

RAID 10 uses logical mirroring to write the same data on two or more drives to provide redundancy. If one disk fails, there is a mirrored image of the data stored on another disk. Additionally, the array uses block-level striping to distribute chunks of data across different drives. This improves performance and read and write speed as the data is simultaneously accessed from multiple disks.

large enterprises for centralizing corporate data. These arrays are designed with a large number of controllers and cache memory.

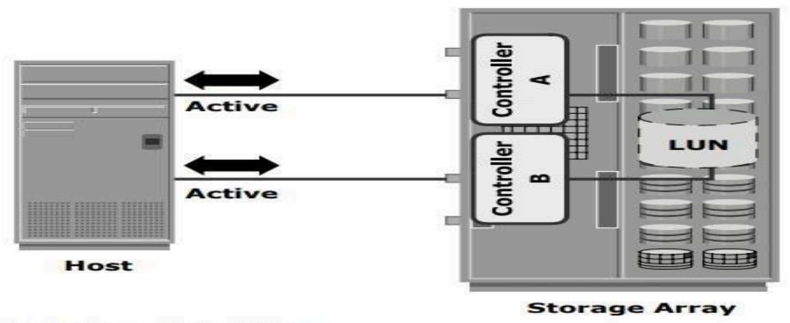


Figure 4-7: Active-active configuration

An active-active array implies that the host can perform I/Os to its LUNs across any of the available paths (see Figure 4-7).

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

- 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

In addition to these features, high-end arrays possess some unique features and functionals that are required for mission-critical applications in large enterprises.

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. 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. As shown in Figure 4-8, 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.

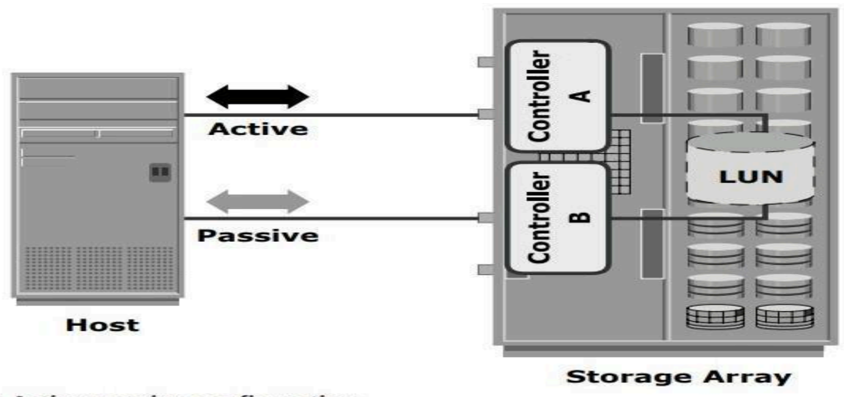


Figure 4-8: Active-passive configuration

Midrange arrays are designed to meet the requirements of small and medium enterprises; therefore, they host less storage capacity and global cache than active-active arrays.

There are also fewer front-end ports for connection to serv- ers. However, they ensure high redundancy and high performance for applications with predictable workloads. They also support array-based local and remote replication.

➔ SCALE-UP AND SCALE OUT STORAGE ARCHITECTURE

- Scaling up and scaling out are the two main methods used to increase data storage capacity.
- Scale-out and scale-up architectures—also known, respectively, as *horizontal scaling* and *vertical scaling* and scale *in* and scale *down*—refer to how companies scale their data storage: by adding more hardware *drives* (scale up/vertical scaling), or by adding more software *nodes* (scale out/horizontal scaling).
- Scale-up is the more traditional format, but it runs into space issues as

data volumes grow and the
need for more and more data storage increases. Hence, the advent of scale-*out* architectures.

- This is a very high-level description of the two main methods of scaling data storage capacity, so let's delve into it a little deeper.

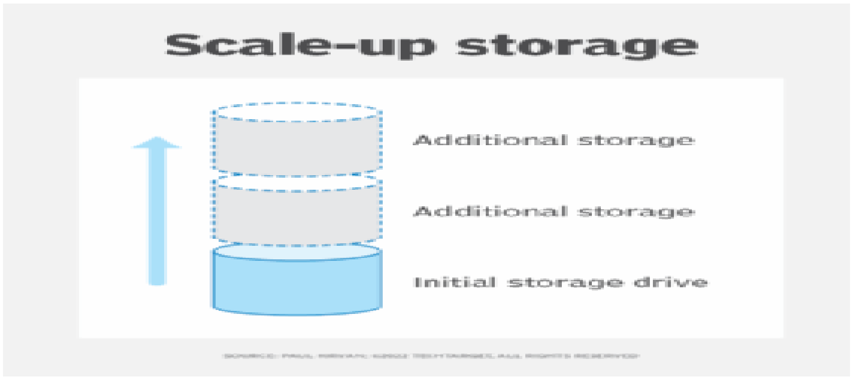
Scale-up Architecture

In a scale-up data storage architecture, storage drives are added to increase storage capacity and performance. The drives are managed by two *controllers*. When you run out of storage capacity, you add another shelf of drives to the architecture.

Scale-up storage and applications

Organizations may need to add capacity to existing storage devices. This could be due to rapid expansion or complexity of one or more applications running on a storage device.
In this type of situation, organizations can increase storage of the specific device. This is referred to as scaling up, as the primary equipment does not change; it only increases its storage capacity.

In a scale-up approach, organizations add to existing infrastructure, such as with more disks or drives. If it is important to retain the same device rather than splitting up critical applications and data across multiple storage devices, use a scale-up approach to scale storage. This is also known as *vertical scaling*.



IT management may determine that an existing storage device will need to increase its capacity due to expansion of key applications that use the storage component. Organizations can then configure additional servers and link them to the main system.

Advantages of Scale-up Architecture

Scaling up offers certain advantages, including:

- **Affordability:** Because there's only one large server to manage, scaling up is a cost-effective way to increase storage capacity since you'll end up paying less for your network equipment and licensing. Upgrading a pre-existing server costs less than purchasing a new one. Vertical scaling also tends to require less new backup and virtualization software.
- **Maintenance:** Since you have only one storage system to manage versus a whole cluster of different elements, scale-up architectures are easier to manage and also make it easier to address specific data quality issues.
- **Simpler communication:** Since vertical scaling means having just a single node handling all the layers of your services, you don't need to worry about your system synchronizing and communicating with other machines to work, which can lead to faster response times.

Disadvantages of Scale-up Architecture

The disadvantages of scale-up architectures include:

- **Scalability limitations:** Although scaling up is how enterprises have traditionally handled storage upgrades, this approach has slowly lost its effectiveness. The RAM, CPU, and hard drives added to a server can only perform to the level the computing housing unit allows. As a result, performance and capacity become a problem as the unit nears its physical limitations. This, in turn, impacts backup and recovery times and other mission-critical processes.
- **Upgrade headaches and downtime:** Upgrading a scale-up architecture can be extremely tedious and involve a lot of heavy lifting.

Typically, you need to copy every piece of data from the old server over to a new machine, which can be costly in terms of both money and downtime. Also, adding another server to the mix usually means adding another data store, which could result in the network getting bogged down by storage pools and users not knowing where to look for files. Both of these can negatively impact productivity. Also, with a scale-up architecture, you need to take your

existing server offline while replacing it with a new, more powerful one. During this time, your apps will be unavailable.

Scale-out Architecture

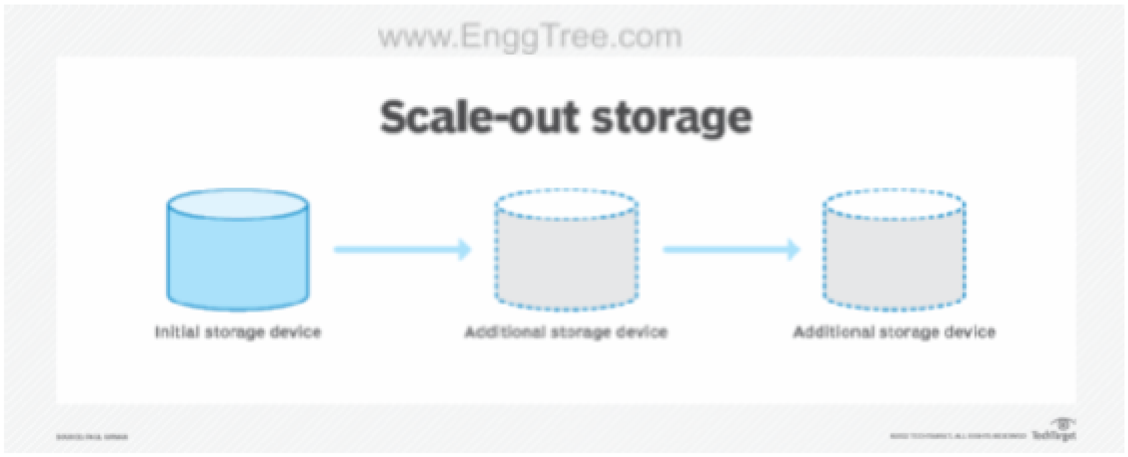
A scale-out architecture uses software-defined storage (SDS) to separate the storage hardware from the storage software, letting the software act as the controllers. This is why scale-out storage is considered to be *network attached storage* (NAS).

Scale-out NAS systems involve clusters of software nodes that work together. Nodes can be added or removed, allowing things like bandwidth, compute, and throughput to increase or decrease as needed. To upgrade a scale-out system, new clusters must be created.

Scale-out storage and applications

For long-term upgrades, management may determine that they need more storage and the unique requirements will need specialized storage devices, such as SSDs and additional HDDs. In practice, it may be necessary to add more equipment racks close to the original storage equipment.

In such situations, it makes more sense to boost storage by configuring a variety of devices that support those requirements. This is referred to as scaling out from the initial storage equipment, or what is also known as *horizontal scaling*.



Distributed file systems can be an important part of a scale-out arrangement, as they use multiple devices in a cohesive storage environment.

Advantages of Scale-out Architecture

The advantages of scale-out architecture include:

- **Better performance:** Horizontal scaling allows for more connection endpoints since the load will be shared by multiple machines, and this improves performance.
- **Easier scaling:** Horizontal scaling is much easier from a hardware perspective because all you need to do is add machines.
- **Less downtime and easier upgrades:** Scaling out means less downtime because you don’t have to switch anything off to scale or make upgrades. Scaling out essentially allows you to upgrade or downgrade your hardware whenever you want as you can move all users, workloads, and data without any downtime. Scale-out systems can also auto-tune and self-heal, allowing clusters to easily accommodate all data demands.

Disadvantages of Scale-out Architecture

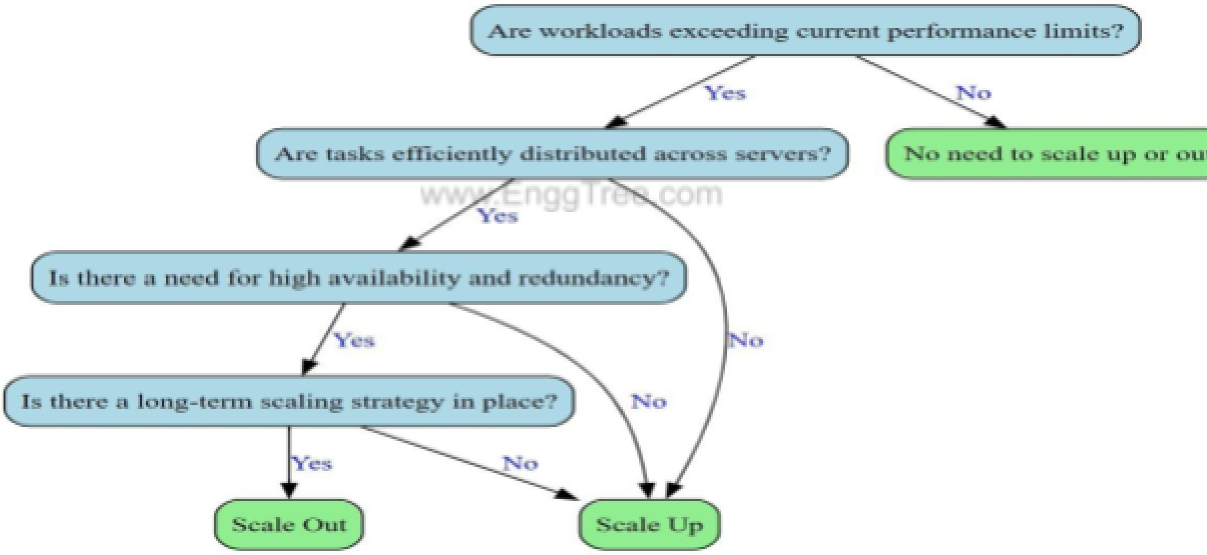
The disadvantages of horizontal scaling include:

- **Complexity:** It’s always going to be harder to maintain multiple servers compared to a single server. Also, things like load balancing and virtualization may require adding software, and machine backups can also be more complex because you'll need to ensure nodes synchronize and communicate effectively.

• **Cost:** Scaling out can be more expensive than scaling up because adding new servers is far more expensive than upgrading old ones.

Scale up or scale out? How to decide

So, should you scale up or scale out your infrastructure? The decision tree below will help you more clearly answer this question.



Are workloads exceeding current performance limits?
Yes
No
Are tasks efficiently distributed across servers?
No need to scale up or ou
Yes
Is there a need for high availability and redundancy?
No
Yes
Is there a long-term scaling strategy in place?
No
Yes
No
Scale Out
Scale Up

Figure A - Decision tree directing whether to scale up or scale out.

Bottom line: Scale up and scale out

Deciding between scaling up and scaling out largely depends on your organization's specific needs and circumstances. Vertical scaling is ideal for situations where a single system can meet the demand, like with high-performance databases.

However, this approach has its limits in terms of hardware capabilities and could lead to higher costs over time.

Conversely, horizontal scaling works best when the workload can be distributed efficiently across multiple servers. This is often preferred for handling web traffic surges or managing user- generated data on platforms like social media sites. Yet, this method can introduce complexities related to managing the distributed system.

In practice, many organizations use a hybrid approach, maximizing each server's power through scaling up, then expanding capacity through scaling out.

Ultimately, the choice between the two strategies should take into account your application's requirements, growth projections and budget. Remember, the goal is to align your scaling strategy with your business objectives for optimal performance.

One great option for scaling your storage is network-attached storage (NAS)