MODULE 2

Data Protection: RAID

- ✓ In 1987, Patterson, Gibson, and Katz at the University of California, Berkeley, published a paper titled "A Case for Redundant Arrays of Inexpensive Disks (RAID)."
- ✓ RAID is the use of small-capacity, inexpensive disk drives as an alternative to large- capacity drives common on mainframe computers.
- ✓ Later RAID has been redefined to refer to independent disks to reflect advances in the storage technology.

2.1 RAID Implementation Methods

The two methods of RAID implementation are hardware and software. Both have their advantages and disadvantages, and are discussed in this section.

2.1.1 Hardware RAID

- ➢ In hardware RAID implementations, a specialized hardware controller is implemented either on the *host* or on the *array*.
- ➢ **Controller card RAID** is a *host-based hardware RAID* implementation in which a specialized RAID controller is installed in the host, and disk drives are connected to it.
- ➢ Manufacturers also integrate RAID controllers on motherboards.
- ➢ A host-based RAID controller is not an efficient solution in a data center environment with a large number of hosts.
- ➢ The external RAID controller is an *array-based hardware RAID*.
- ➢ It acts as an interface between the host and disks.
- ➢ It presents storage volumes to the host, and the host manages these volumes as physical drives.
- ➢ The key functions of the RAID controllers are as follows:
  - ✓ Management and control of disk aggregations
  - ✓ Translation of I/O requests between logical disks and physical disks
  - ✓ Data regeneration in the event of disk failures

2.1.2 Software RAID

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

- ✓ **Performance:** Software RAID affects overall system performance. This is due to additional CPU cycles required to perform RAID calculations.
- ✓ **Supported features:** Software RAID does not support all RAID levels.
- ✓ **Operating system compatibility:** Software RAID is tied to the host operating system; hence, upgrades to software RAID or to the operating system should be validated for compatibility. This leads to inflexibility in the data-processing environment.

2.2 RAID Array Components

- A RAID array is an enclosure that contains a number of disk drives and supporting hardware to implement RAID.
- A subset of disks within a RAID array can be grouped to form logical associations called logical arrays, also known as a RAID set or a RAID group (see Figure 2-1).

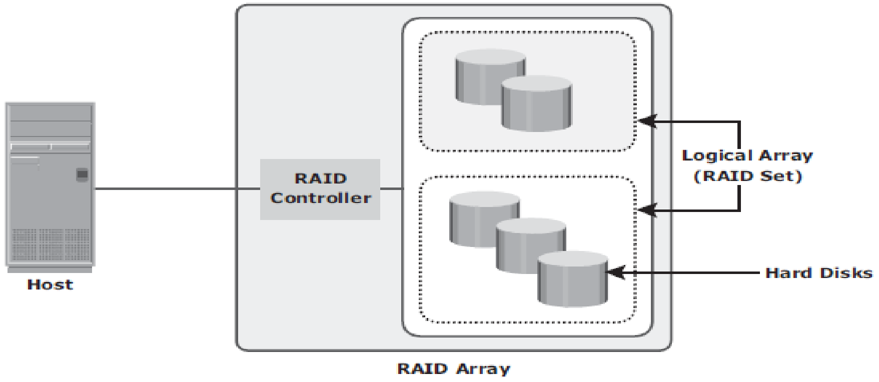


Figure 2-1: Components of a RAID array

2.3 RAID Techniques

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

These techniques determine the data availability and performance characteristics of a RAID set.

2.3.1 Striping

- **Striping** is a technique to spread data across multiple drives (more than one) to use the drives in parallel.
- All the read-write heads work simultaneously, allowing more data to be processed in a shorter time and increasing performance, compared to reading and writing from a single disk.
- Within each disk in a RAID set, a **predefined number of contiguously addressable** disk blocks are defined as a **strip**.
- The set of aligned strips that spans across all the disks within the RAID set is called a **stripe**.
- Fig 2-2 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 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

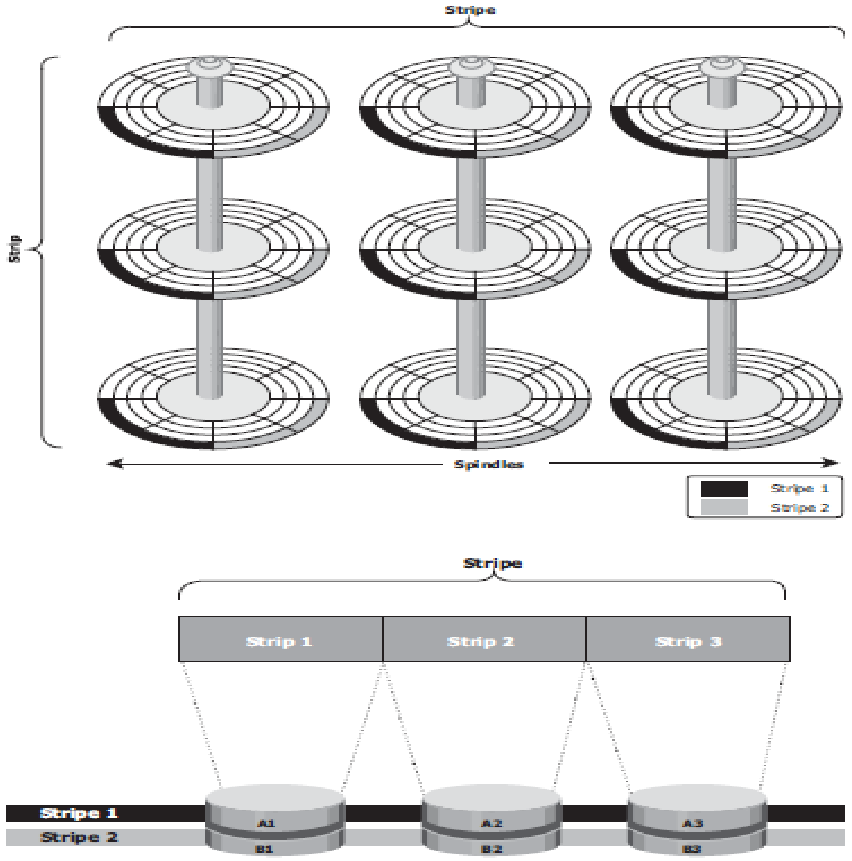


Figure 2-2: Striped RAID set

2.3.2 Mirroring

- **Mirroring** is a technique whereby the same data is stored on two different disk drives, yielding two copies of the data.
- If one disk drive failure occurs, the data is intact on the surviving disk drive (see Fig 2-3) 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

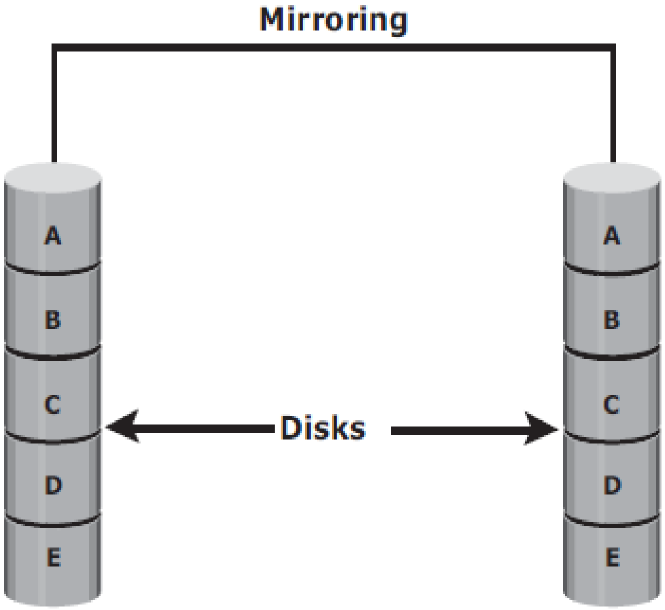


Figure 2-3: Mirrored disks in an array

2.3.3 Parity

- **Parity** is a method to protect striped data from disk drive failure without the cost of mirroring.
- *An additional disk drive is added to hold parity*, a mathematical construct that allows re-creation of the missing data.
- Parity is a **redundancy technique** that ensures protection of data without maintaining a full set of duplicate data.
- Calculation of parity is a function of the RAID controller.
- Parity information can be stored on separate, dedicated disk drives or distributed across all the drives in a RAID set.
- Fig 2-4 shows a parity RAID set.

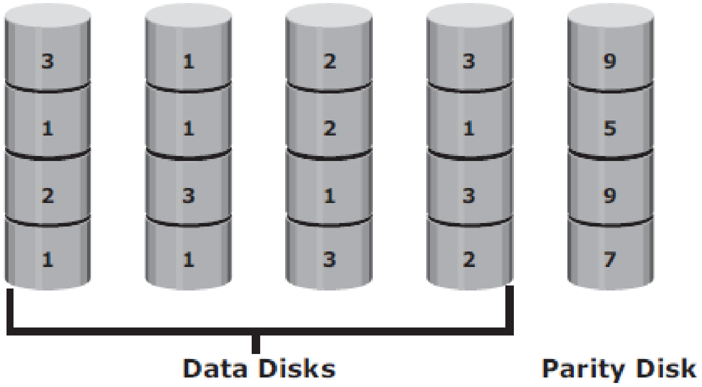


Fig 2-4: Parity RAID

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

XOR Operation:

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

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

Table 1.1: Truth table for XOR Operation

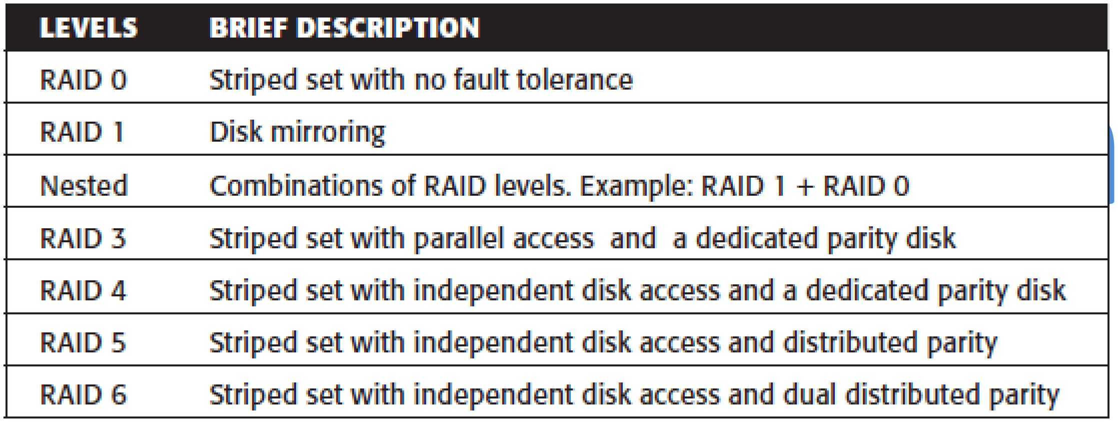
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

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

2.4 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 2.1 shows the commonly used RAID levels

Table 2.1: 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

2.4.1 RAID 0

- **RAID 0** configuration uses *data striping techniques,* where data is striped across all the disks within a RAID set. Therefore it utilizes the full storage capacity of a RAID set.
- To read data, all the strips are put back together by the controller.
- Fig 2-5 shows RAID 0 in an array in which data is striped across five disks.

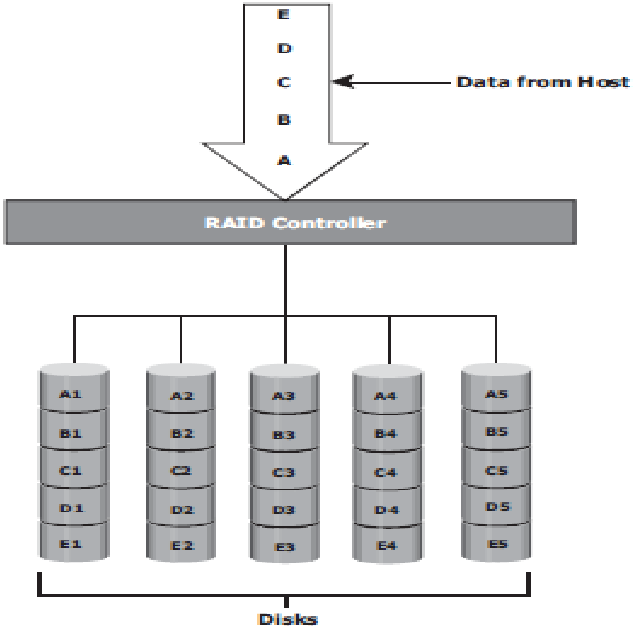


Fig 2-5: RAID 0

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

2.4.2 RAID 1

- **RAID 1** is based on the *mirroring* technique.
- In this RAID configuration, data is mirrored to provide *fault tolerance* (see Fig 2-6). A
- RAID 1 set consists of two disk drives and every write is written to both disks.
- The mirroring is transparent to the host.
- During disk failure, the impact on data recovery in RAID 1 is the least among all RAID implementations. This is because the RAID controller uses the mirror drive for data recovery.
- RAID 1 is suitable for applications that require high availability and cost is no constraint.

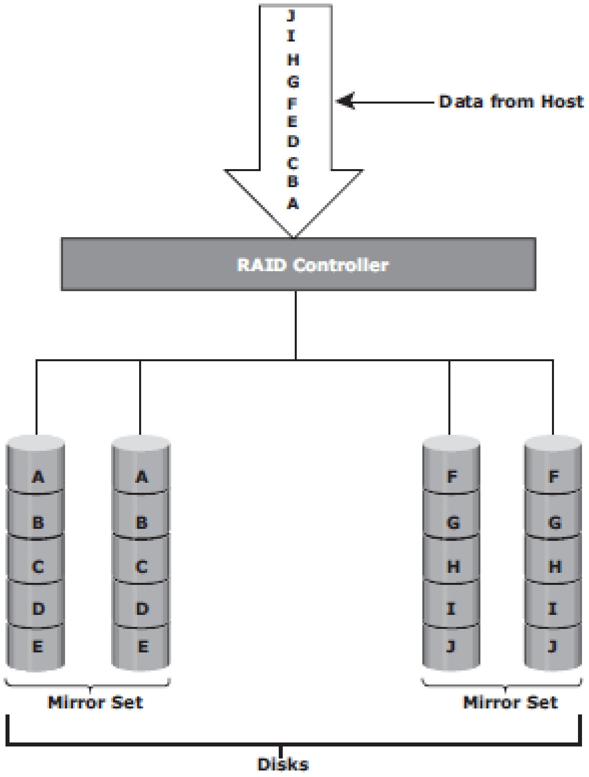


Fig 2-6: RAID 1

2.4.3 Nested RAID

- Most data centers require data redundancy and performance from their RAID arrays.
- RAID 1+0 and RAID 0+1 combine the performance benefits of RAID 0 with the redundancy benefits of RAID 1.
- They use striping and mirroring techniques and combine their benefits.
- These types of RAID require an even number of disks, the minimum being four (see Fig 2-7).

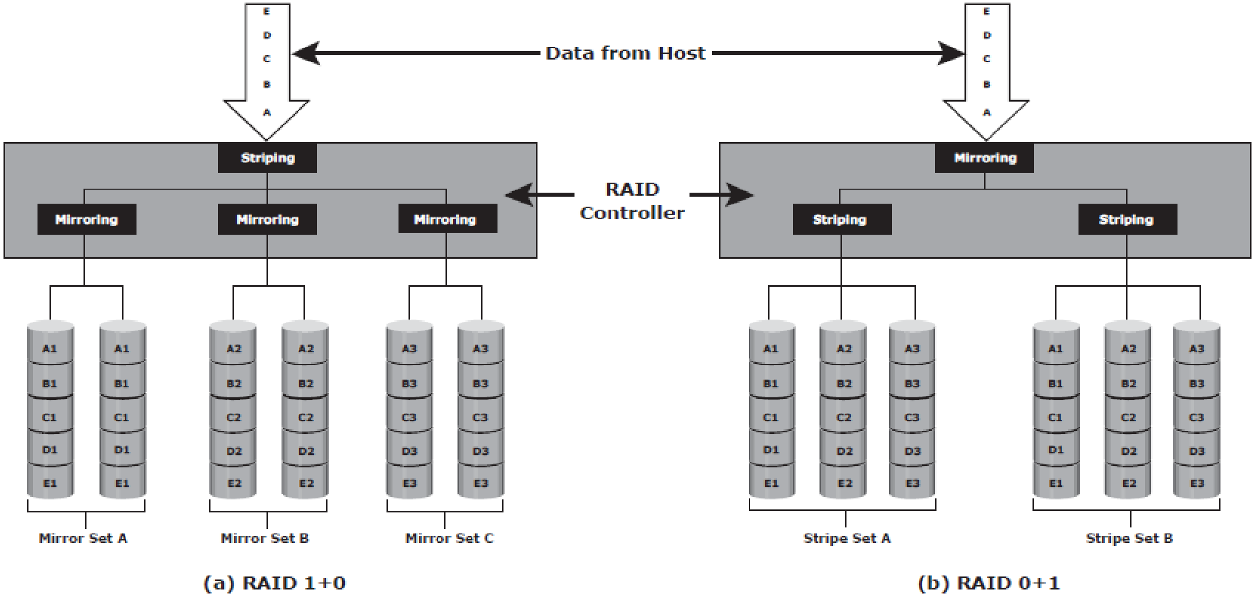


Fig 2-7: Nested RAID

<u>RAID 1+0:</u>

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

<u>Working of RAID 1+0:</u>

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

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

  <u>Working of RAID 0+1:</u>
- Eg: Consider the same example of six disks forming a RAID 0+1 (that is, RAID 0 first and then RAID 1).
- Here, six disks are paired into two sets of three disks each.
- Each of these sets, in turn, act as a RAID 0 set that contains three disks and then these two sets are mirrored to form RAID 1.
- Following are the steps performed in RAID 0+1 (see Fig 1.16 [b]):
  - ✓ Drives 1 + 2 + 3 = RAID 0 (Stripe Set A)
  - ✓ Drives 4 + 5 + 6 = RAID 0 (Stripe Set B)
- These two stripe sets are mirrored.
- If one of the drives, say drive 3, fails, the entire stripe set A fails.
- A rebuild operation copies the entire stripe, copying the data from each disk in the healthy stripe to an equivalent disk in the failed stripe.
- This causes increased and unnecessary I/O load on the surviving disks and makes the RAID set more vulnerable to a second disk failure.

2.4.4 RAID 3

- RAID 3 stripes data for high performance and uses parity for improved fault tolerance.
- Parity information is stored on a dedicated drive so that data can be reconstructed if a drive fails. For example, of five disks, four are used for data and one is used for parity.
- RAID 3 always reads and writes complete stripes of data across all disks, as the drives operate in parallel. There are no partial writes that update one out of many strips in a stripe.
- RAID 3 provides good bandwidth for the transfer of large volumes of data. RAID 3 is used in applications that involve large sequential data access, such as video streaming.
- Fig 2-8 shows the RAID 3 implementation

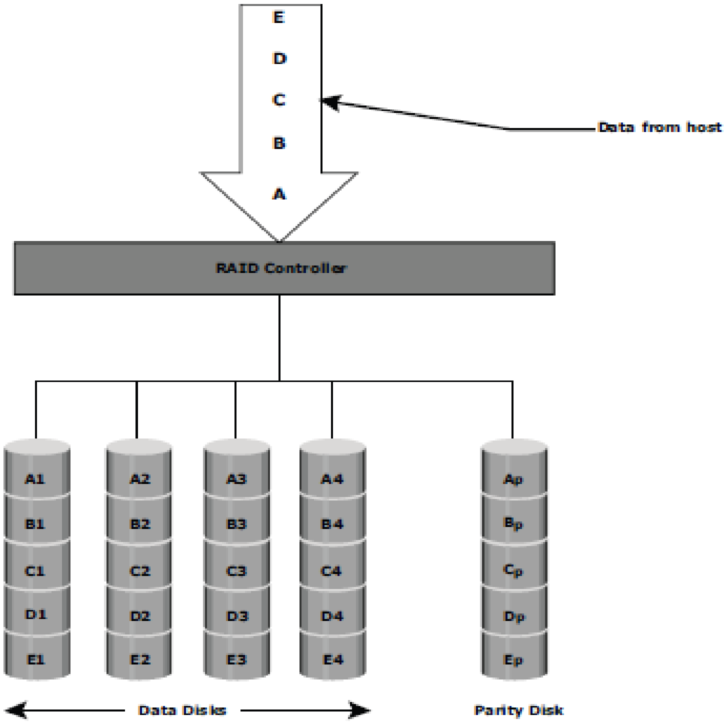


Fig 2-8: RAID 3

2.4.5 RAID 4

- RAID 4 stripes data for high performance and uses parity for improved fault tolerance. Data is striped across all disks except the parity disk in the array.
- Parity information is stored on a dedicated disk so that the data can be rebuilt if a drive fails. Striping is done at the block level.

- Unlike RAID 3, data disks in RAID 4 can be accessed independently so that specific data elements can be read or written on single disk without read or write of an entire stripe. RAID 4 provides good read throughput and reasonable write throughput.

2.4.6 RAID 5

- RAID 5 is a versatile RAID implementation.
- It is similar to RAID 4 because it uses striping. The drives (strips) are also independently accessible.
- The difference between RAID 4 and RAID 5 is the parity location. In RAID 4, parity is written to a dedicated drive, creating a write bottleneck for the parity disk
- In RAID 5, parity is distributed across all disks. The distribution of parity in RAID 5 overcomes the Write bottleneck. Below Figure illustrates the RAID 5 implementation.
- Fig 2-9 illustrates the RAID 5 implementation.
- RAID 5 is good for random, read-intensive I/O applications and preferred for messaging, data mining, medium-performance media serving, and relational database management system (RDBMS) implementations, in which database administrators (DBAs) optimize data access.

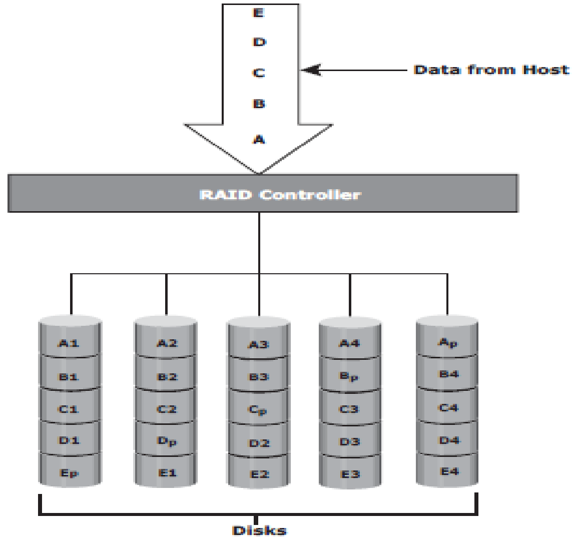


Fig 2-9: RAID 5

2.4.7 RAID 6

- RAID 6 includes a second parity element to enable survival in the event of the failure of two disks in a RAID group. Therefore, a RAID 6 implementation requires at least four disks.
- RAID 6 distributes the parity across all the disks. The write penalty in RAID 6 is more than that in RAID 5; therefore, RAID 5 writes perform better than RAID 6. The rebuild operation in RAID 6 may take longer than that in RAID 5 due to the presence of two parity sets.
- Fig 2-10 illustrates the RAID 6 implementation

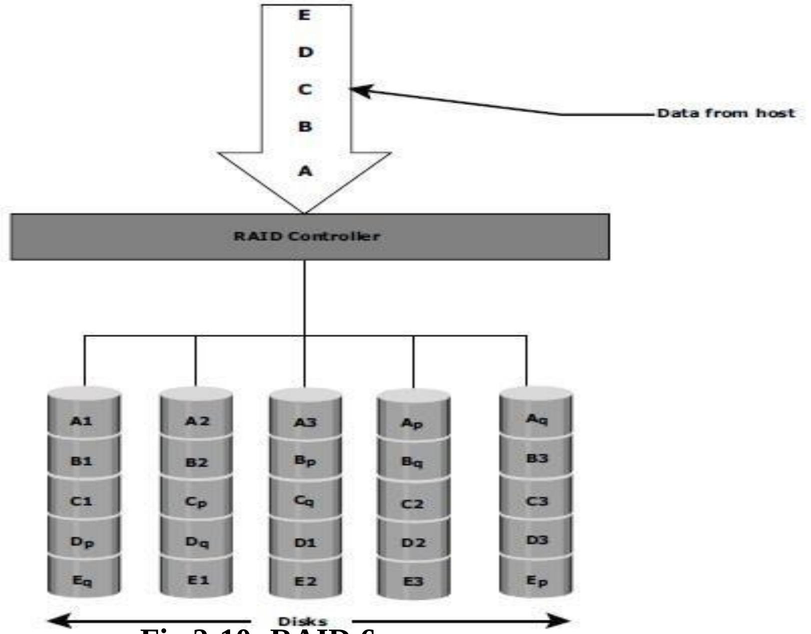


Fig 2-10: RAID 6

2.5 RAID Impact on Disk Performance

- When choosing a RAID type, it is imperative to consider its impact on disk performance and application IOPS.
- In both mirrored (RAID 1) and parity RAID (RAID 5) configurations, every write operation translates into more I/O overhead for the disks which is referred to as **write penalty**.
- In a RAID 1 implementation, every write operation must be performed on two disks configured as a mirrored pair. **The write penalty is 2.**
- In a RAID 5 implementation, a write operation may manifest as four I/O operations. When performing small I/Os to a disk configured with RAID 5, the controller has to read, calculate, and write a parity segment for every data write operation.
- Fig 2-11 illustrates a single write operation on RAID 5 that contains a group of five disks.

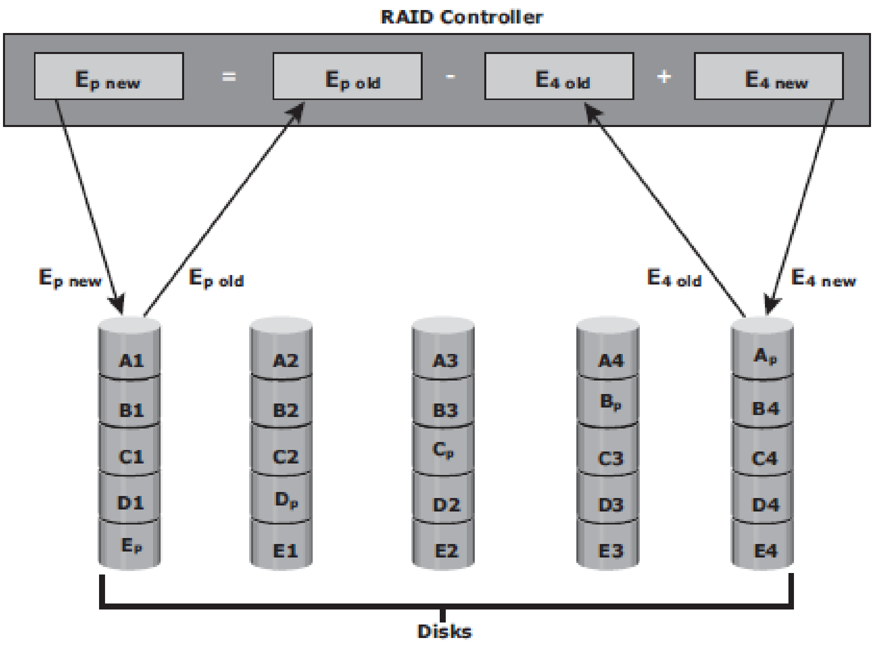


Fig 2-11: Write Penalty in RAID 5

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

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

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

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

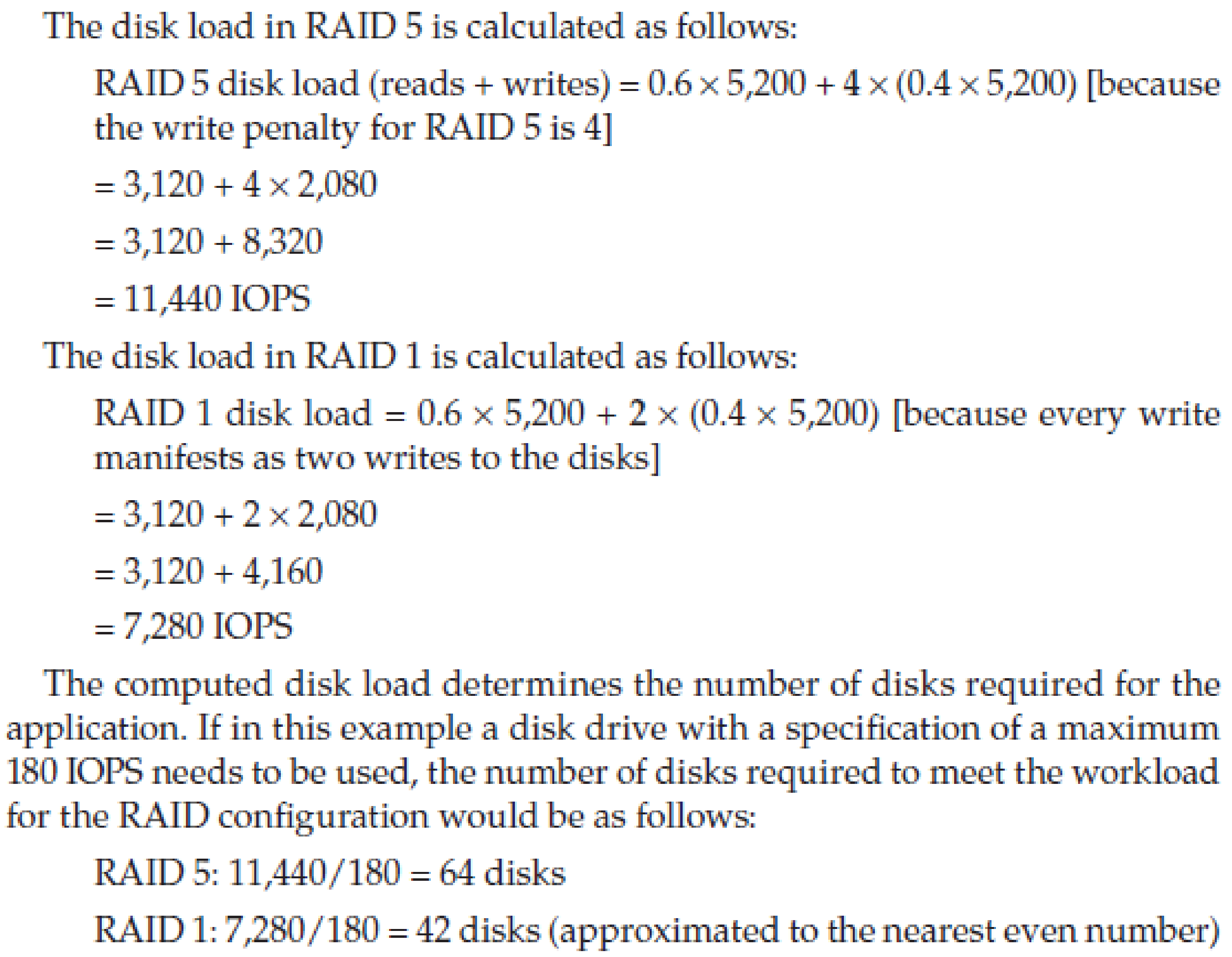
- After computing the new parity, the controller completes the write I/O by doing two write I/Os for the new data and the new parity onto the disks..
- Therefore, the controller performs two disk reads and two disk writes for every write operation, and **the write penalty is 4**.
- In RAID 6, which maintains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

2.5.1 Application IOPS and RAID Configurations

When deciding the number of disks required for an application, it is important to consider the impact of RAID based on IOPS generated by the application. The total disk load should be computed by considering the type of RAID configuration and the ratio of read compared to write from the host.

- The following example illustrates the method to compute the disk load in different types of RAID.

Consider an application that generates 5,200 IOPS, with 60 percent of them being reads.



The disk load in RAID 5 is calculated as follows:

RAID 5 disk load (reads + writes) = 0.6 × 5,200 + 4 × (0.4 × 5,200) [because the write penalty for RAID 5 is 4]

= 3,120 + 4 × 2,080

= 3,120 + 8,320

= 11,440 IOPS

The disk load in RAID 1 is calculated as follows:

RAID 1 disk load = 0.6 × 5,200 + 2 × (0.4 × 5,200) [because every write manifests as two writes to the disks]

= 3,120 + 2 × 2,080

= 3,120 + 4,160

= 7,280 IOPS

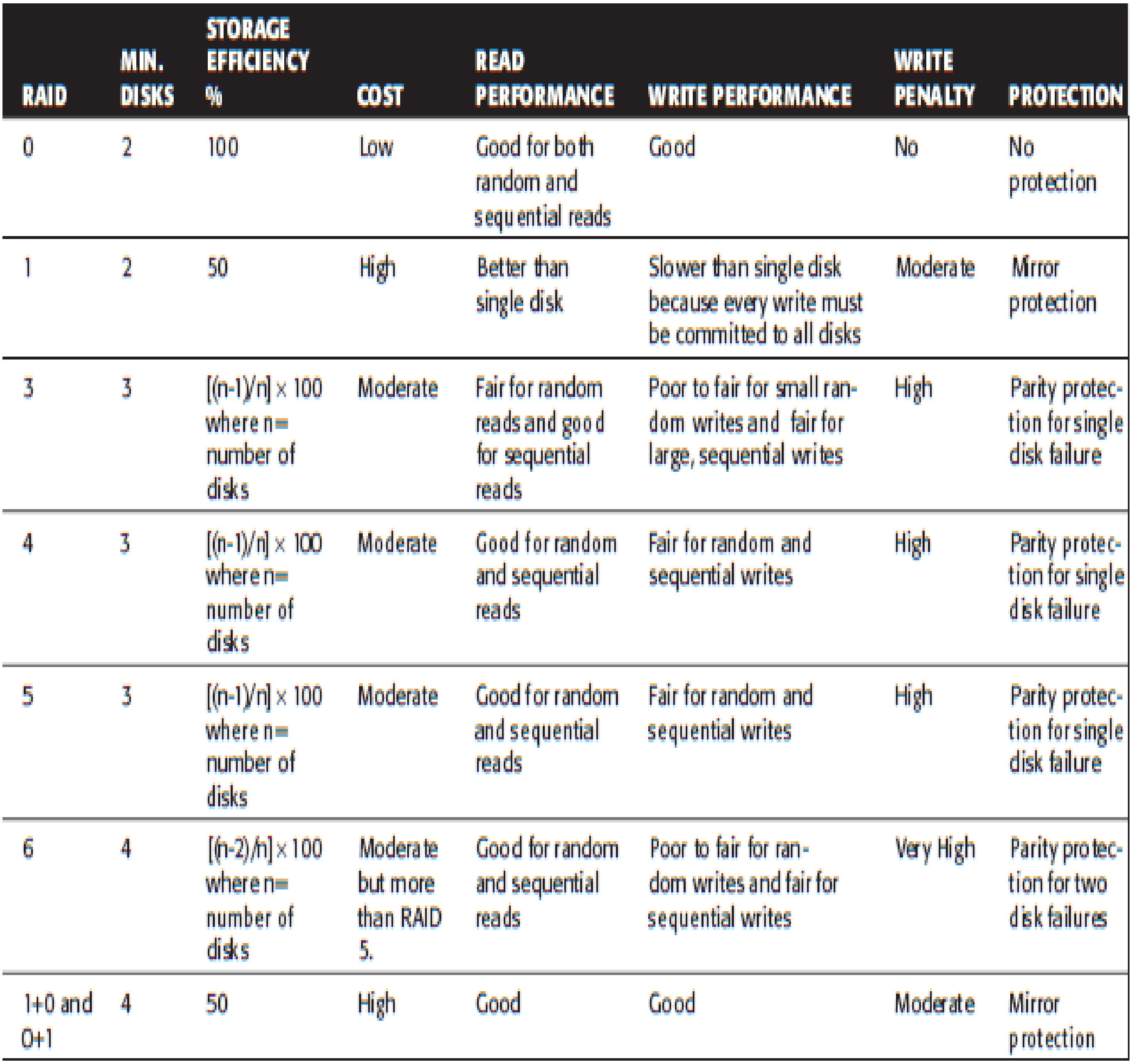
The computed disk load determines the number of disks required for the application. If in this example a disk drive with a specification of a maximum 180 IOPS needs to be used, the number of disks required to meet the workload for the RAID configuration would be as follows:

RAID 5: 11,440/180 = 64 disks

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

2.5 RAID Comparison

Table 2-2 compares the common types of RAID levels.



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

CHAPTER-2 MODULE 2-Intelligent Storage Systems

4.1 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 4-1):
  - ✓ Front End
  - ✓ Cache
  - ✓ Back end
  - ✓ Physical disks.
- An I/O request received from the host at the front-end port is processed through cache and the back end, to enable storage and retrieval of data from the physical disk.
- A read request can be serviced directly from cache if the requested data is found in cache.
- In modern intelligent storage systems, front end, cache, and back end are typically integrated on a single board (referred to as a storage processor or storage controller).

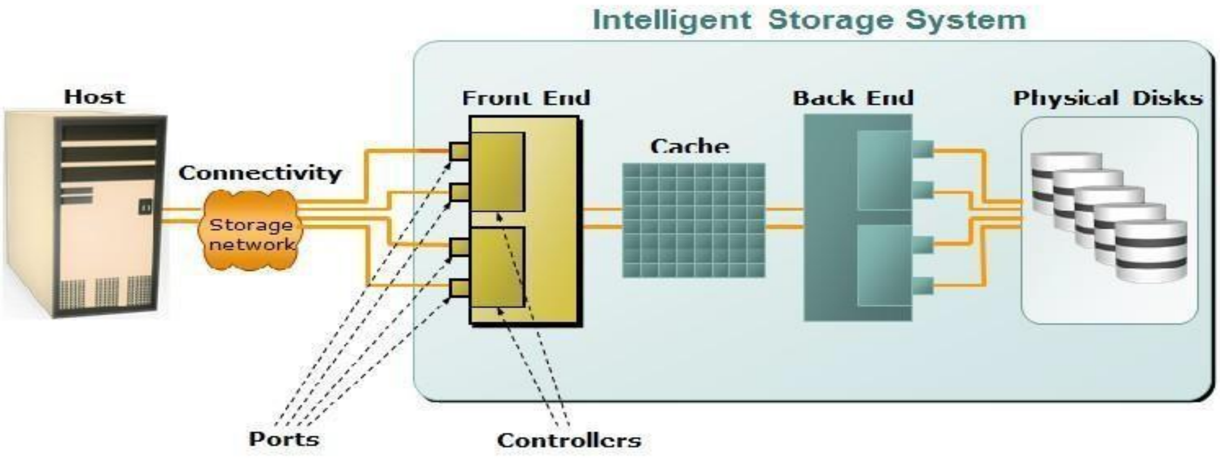


Fig 4-1 Components of an Intelligent Storage System

4.1.1 Front End

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

  i. Front-End Ports ii. Front-end controllers

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

4.1.2 Cache

- **Cache** is semiconductor memory where data is placed temporarily to reduce the time required to service I/O requests from the host.
- Cache improves storage system **performance** by isolating hosts from the mechanical delays associated with rotating disks or hard disk drives (HDD).
- Rotating disks are the slowest component of an intelligent storage system. Data access on rotating disks usually takes several millisecond because of seek time and rotational latency.
- **Accessing data from cache is fast and typically takes less than a millisecond**.
- On intelligent arrays, write data is first placed in cache and then written to disk.

Structure Of Cache

- Cache is organized into pages, which is the smallest unit of cache allocation. The size of a cache page is configured according to the application I/O size.
- Cache consists of the **data store** and **tag RAM**.
- The data store holds the data whereas the tag RAM tracks the location of the data in the data store (see Fig 4-2) and in the disk.
- Entries in tag RAM indicate where data is found in cache and where the data belongs on the disk.
- Tag RAM includes a dirty bit flag, which indicates whether the data in cache has been committed to the disk.
- It also contains time-based information, such as the time of last access, which is used to identify cached information that has not been accessed for a long period and may be freed up.

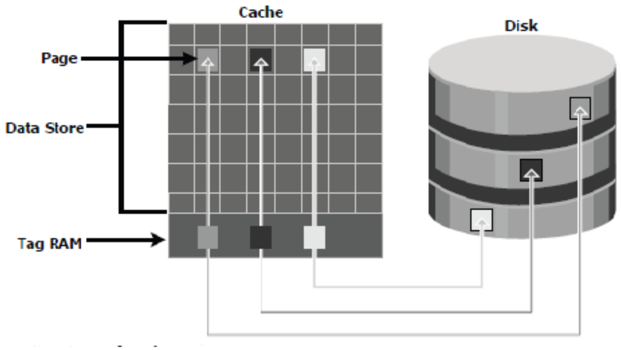


Fig 4-2: Structure of cache

Read Operation with Cache

- When a host issues a read request, the storage controller reads the tag RAM to determine whether the required data is available in cache.
- If the requested data is found in the cache, it is called a **read cache hit** or **read hit** and data is sent directly to the host, without any disk operation (see Fig 4-2 [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 Fig 4-2 [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 thefront- 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.

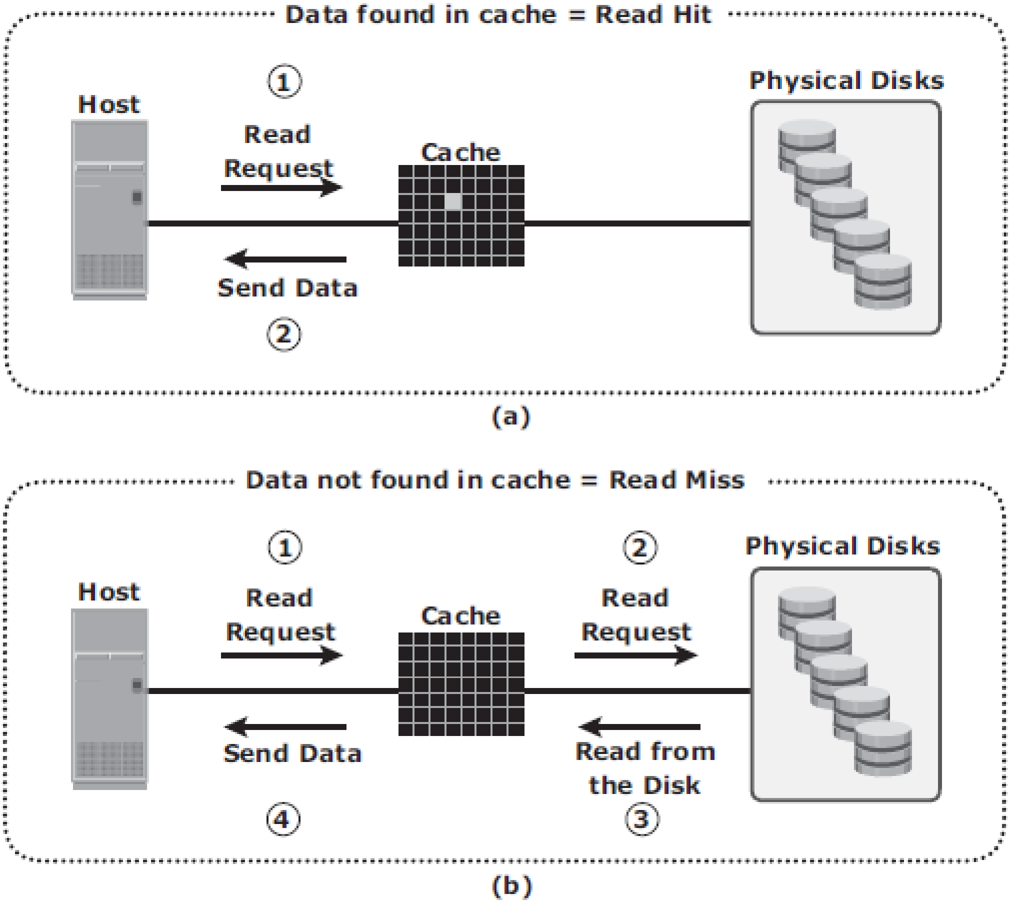


Figure 4-3: Read hit and read miss

Write Operation with Cache

- Write operations with cache provide performance advantages over writing directly to disks.
- When an I/O is written to cache and acknowledged, it is completed in far less time (from the host's perspective) than it would take to write directly to disk.
- *Sequential writes* also offer opportunities for optimization because many smaller writes can be coalesced for larger transfers to disk drives with the use of cache.
- **A write operation** with cache is implemented in the following ways:
- **Write-back cache:** Data is placed in cache and an acknowledgment is sent to the host immediately. Later, data from several writes are committed to the disk. Write response times are much faster, as the write operations are isolated from the mechanical delays of the disk. However, uncommitted data is at risk of loss in the event of cache failures.
- **Write-through cache**: Data is placed in the cache and immediately written to the disk, and an acknowledgment is sent to the host. Because data is committed to disk as it arrives,

the risks of data loss are low but write response time is longer because of the disk operations.

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

Cache Implementation

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

Cache Management

- Cache is a finite and expensive resource that needs proper management.
- Even though modern intelligent storage systems come with a large amount of cache, when all cache pages are filled, some pages have to be freed up to accommodate new data and avoid performance degradation.
- Various cache management algorithms are implemented in intelligent storage systems to proactively maintain a set of free pages and a list of pages that can be potentially freed up whenever required.
- The most commonly used algorithms are listed below:
  - ✓ **Least Recently Used (LRU):** An algorithm that continuously monitors data access in cache and identifies the cache pages that have not been accessed for a long time. LRU either frees up these pages or marks them for reuse. This algorithm is based on the assumption that data which hasn’t been accessed for a while will not be requested by the host.

- ✓ **Most Recently Used (MRU):** In MRU, the pages that have been accessed most recently are freed up or marked for reuse. This algorithm is based on the assumption that recently accessed data may not be required for a while

- As cache fills, the storage system must take action to **flush dirty pages** (data written into the cache but not yet written to the disk) to manage space availability.
- **Flushing** is the process that commits data from cache to the disk.
- On the basis of the I/O access rate and pattern, high and low levels called **watermarks** are set in cache to manage the flushing process.
- **High watermark (HWM)** is the cache utilization level at which the storage system starts high-speed flushing of cache data.
- **Low watermark (LWM)** is the point at which the storage system stops flushing data to the disks.
- The *cache utilization level*, as shown in Fig 4-4, drives the mode of flushing to be used:
  - ✓ **Idle flushing:** Occurs continuously, at a modest rate, when the cache utilization level is between the high and low watermark.
  - ✓ **High watermark flushing:** Activated when cache utilization hits the high watermark. The storage system dedicates some additional resources for flushing. This type of flushing has some impact on I/O processing.
  - ✓ **Forced flushing:** Occurs in the event of a large I/O burst when cache reaches 100 percent of its capacity, which significantly affects the I/O response time. In forced flushing, system flushes the cache on priority by allocating more resources.

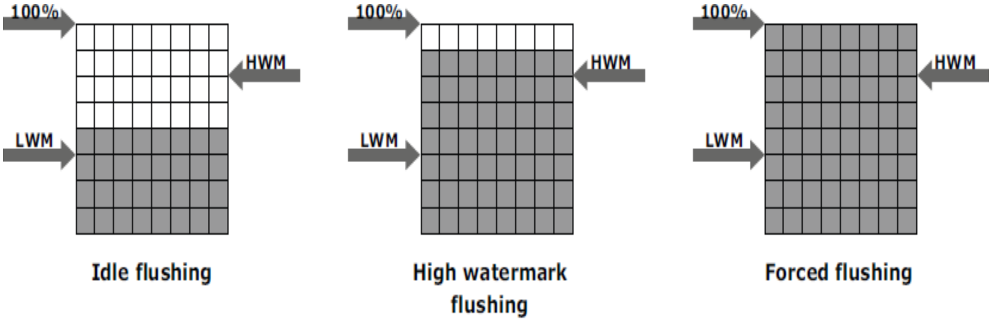


Fig 4-4: Types of flushing

Cache Data Protection

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

- This risk of losing uncommitted data held in cache can be mitigated using
- **Cache mirroring**
  - ✓ Each write to cache is held in two different memory locations on two independent memory cards. In the event of a cache failure, the write data will still be safe in the mirrored location and can be committed to the disk.
  - ✓ Reads are staged from the disk to the cache, therefore, in the event of a cache failure, the data can still be accessed from the disk.
  - ✓ In cache mirroring approaches, the problem of maintaining *cache coherency* is introduced.
  - ✓ Cache coherency means that data in two different cache locations must be identical at all times. It is the responsibility of the array operating environment to ensure coherency.
- **Cache vaulting**
  - ✓ The risk of data loss due to power failure can be addressed in various ways:
    - powering the memory with a battery until the AC power is restored
    - using battery power to write the cache content to the disk.
  - ✓ If an extended power failure occurs, using batteries is not a viable option.
  - ✓ This is because in intelligent storage systems, large amounts of data might need to be committed to numerous disks, and batteries might not provide power for sufficient time to write each piece of data to its intended disk.
  - ✓ Storage vendors use a set of physical disks to dump the contents of cache during power failure. This is called *cache vaulting* and the disks are called vault drives.
  - ✓ When power is restored, data from these disks is written back to write cache and then written to the intended disks.

4.1.3 Back End

- The **back end** provides an interface between cache and the physical disks.
- It consists of two components:

- Back-end ports
- Back-end controllers.

- The back end controls data transfers between cache and the physical disks.
- From cache, data is sent to the back end and then routed to the destination disk.

- Physical disks are connected to *ports* on the back end.
- The *back end controller* communicates with the disks when performing reads and writes and also provides additional, but limited, temporary data storage.
- The algorithms implemented on back-end controllers provide error detection and correction, and also RAID functionality.
- For high data protection and high availability, storage systems are configured with dual controllers with multiple ports.

4.1.4 Physical Disk

- A physical disk stores data persistently.
- Physical disks are connected to the back-end storage controller and provide persistent data storage.
- Modern intelligent storage systems provide support to a variety of disk drives with different speeds and types, such as FC, SATA, SAS, and flash drives.
- They also support the use of a mix of flash, FC, or SATA within the same array.

4.2 Types of Intelligent Storage Systems

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

4.2.1 High-end Storage Systems

- High-end storage systems, referred to as **active-active arrays,** are generally aimed at large enterprises for centralizing corporate data. These arrays are designed with a large number of controllers and cache memory.
- An active-active array implies that the host can perform I/Os to its LUNs across any of the available paths (see Fig 4-5).

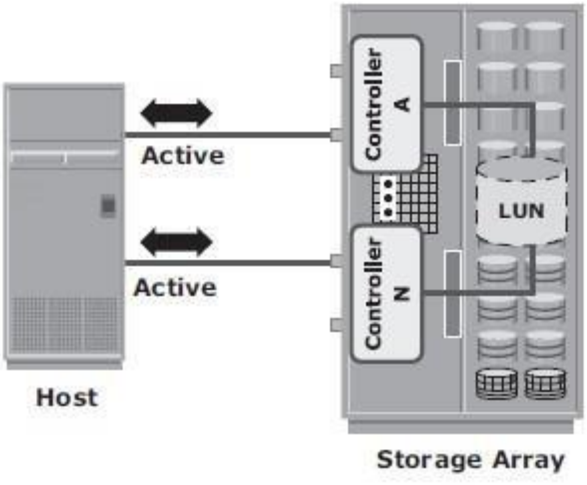


Fig 4-5: Active-active configuration

Advantages of High-end storage:

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

4.2.2 Midrange Storage System

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

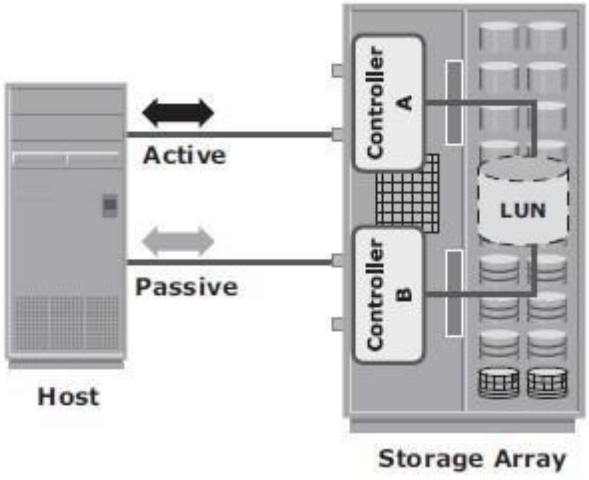


Fig 4-6: Active-passive configuration

- As shown in Fig 4-6, the host can perform reads or writes to the LUN only through the path to controller A, as controller A is the owner of that LUN.
- The path to controller B remains **Passive** and no I/O activity is performed through this path.
- Midrange storage systems are typically designed with two controllers, each of which contains host interfaces, cache, RAID controllers, and disk drive interfaces.
- Midrange arrays are designed to meet the requirements of small and medium enterprise applications; therefore, they host less storage capacity and cache than high-end storage arrays.
- There are also fewer front-end ports for connection to hosts.
- But they ensure high redundancy and high performance for applications with predictable workloads.
- They also support array-based local and remote replication.

CHAPTER-3 MODULE 2-Fibre Channel Storage Area Networks

SAN is a high-speed dedicated network of servers and shared storage. Common SAN deployments are:

- ✓ FC SAN
- ✓ IP SAN

5.1 Fibre Channel: Overview

- The FC architecture forms the fundamental construct of the SAN infrastructure.
- **Fibre Channel** is a high-speed network technology that runs on high-speed optical fiber cables (preferred for front-end SAN connectivity) and serial copper cables (preferred for back-end disk connectivity).
- The FC technology was created to meet the demand for increased speeds of data transfer among computers, servers, and mass storage subsystems.

5.2The SAN and Its Evolution

- A SAN carries data between servers (or hosts) and storage devices through Fibre Channel network (see Figure 5-1).
- A SAN enables storage consolidation and enables storage to be shared across multiple servers. This improves the utilization of storage resources compared to direct-attached storage architecture and reduces the total amount of storage an organization needs to purchase and manage.
- With consolidation, storage management becomes centralized and less complex, which further reduces the cost of managing information. SAN also enables organizations to connect geographically dispersed servers and storage.

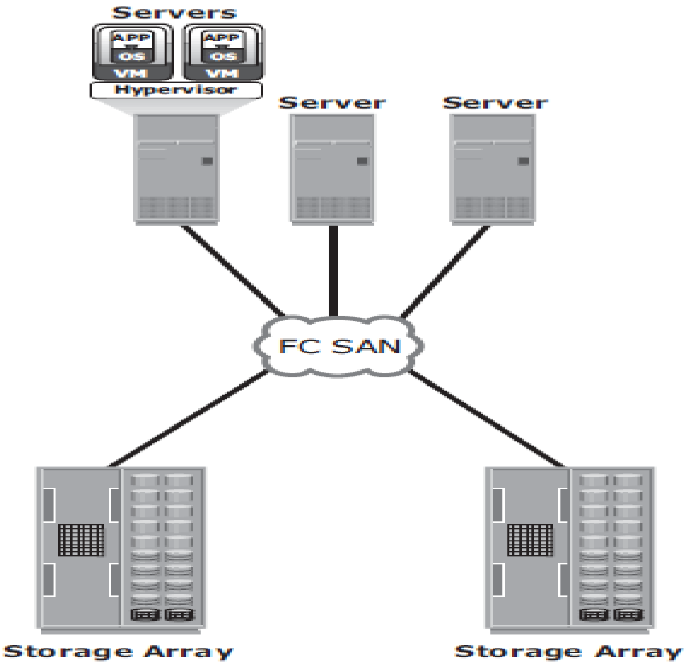


Figure 5-1: FC SAN implementation

- In its earliest implementation, the FC SAN was a simple grouping of hosts and storage devices connected to a network using an FC hub as a connectivity device. This configuration of an FC SAN is known as a Fibre Channel Arbitrated Loop (FC-AL). Use of hubs resulted in isolated FC-AL SAN islands because hubs provide limited connectivity and bandwidth.
- The inherent limitations associated with hubs gave way to high-performance FC switches. Use of switches in SAN improved connectivity and performance and enabled FC SANs to be highly scalable. Figure 5-2 illustrates the FC SAN evolution from FC-AL to enterprise SANs.

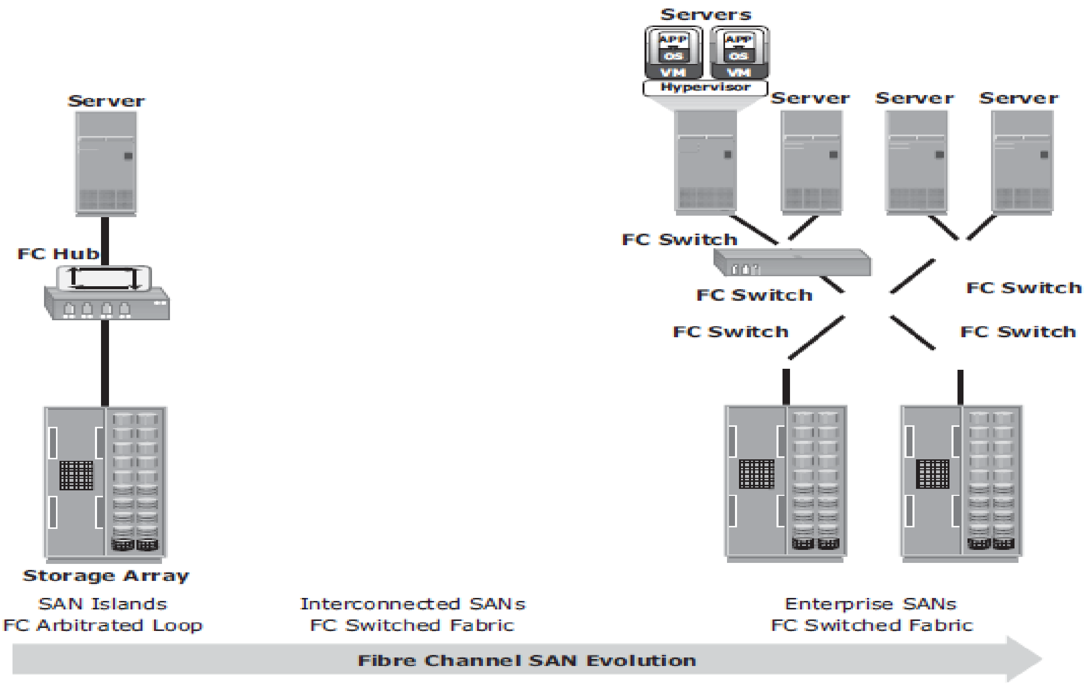


Figure 5-2: FC SAN evolution

5.3 Components of SAN

- Components of FC SAN infrastructure are:

1) **Node Ports,**
2) **Cabling,**
3) **Connectors,**
4) **Interconnecting Devices (Such as FC Switches or Hubs),**
5) **San Management Software**.

5.3.1 Node Ports

- In fibre channel, devices such as hosts, storage and tape libraries are all referred to as **Nodes.**
- Each node is a **source or destination** of information for one or more nodes.
- Each node requires one or more ports to provide a physical interface for communicating with other nodes.
- A port operates in full-duplex data transmission mode with a **transmit (Tx) link and a**

receive (Rx) link (see Fig 5-3).

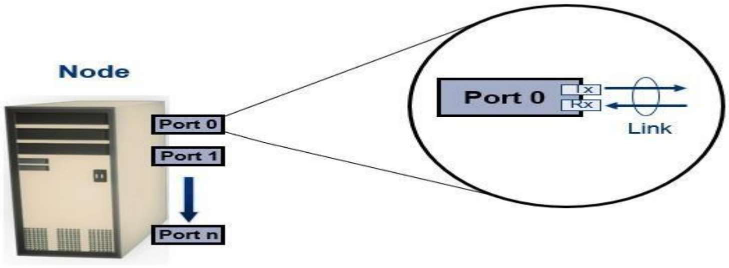


Fig 5-3: Nodes, Ports, links

5.3.2 Cables and Connectors

Cabling

- SAN implementations use optical fiber cabling.
- Copper can be used for shorter distances for back-end connectivity
- Optical fiber cables carry data in the form of light.
- There are two types of optical cables: **Multi-Mode And Single-Mode.**

1) **Multi-mode fiber (MMF)** cable carries multiple beams of light projected at different angles simultaneously onto the core of the cable (see Fig 5-4(a)).
    - In an MMF transmission, multiple light beams traveling inside the cable tend to disperse and collide. This collision weakens the signal strength after it travels a certain distance — a process known as *modal dispersion*.
    - MMFs are generally used within data centers for shorter distance runs

2) **Single-mode fiber (SMF)** carries a single ray of light projected at the center of the core (see Fig 5-4(b)).
    - In an SMF transmission, a single light beam travels in a straight line through the core of the fiber.
    - The small core and the single light wave limits modal dispersion. Among all types of fibre cables, single-mode provides minimum signal attenuation over maximum distance (up to 10 km).
    - A single-mode cable is used for long-distance cable runs, limited only by the power

of the laser at the transmitter and sensitivity of the receiver.

- SMFs are used for longer distances.

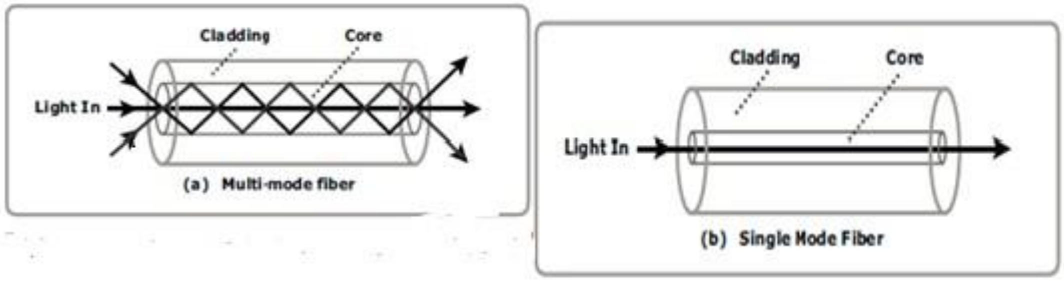


Fig 5-4: Multimode fiber and single-mode fiber

<u>Connectors</u>

- They are attached at the end of the cable to enable swift connection and disconnection of the cable to and from a port.
- A **Standard connector (SC)** (see Fig 5-5(a)) and a **Lucent connector (LC)** (see Fig 5-5(b)) are two commonly used connectors for fiber optic cables.
- An SC is used for data transmission speeds up to 1 Gb/s, whereas an LC is used for speeds up to 4 Gb/s.
- Figure 2.3 depicts a Lucent connector and a Standard connector.
- A Straight Tip (ST) is a fiber optic connector with a plug and a socket that is locked with a half-twisted bayonet lock (see Fig 5-5(c)).

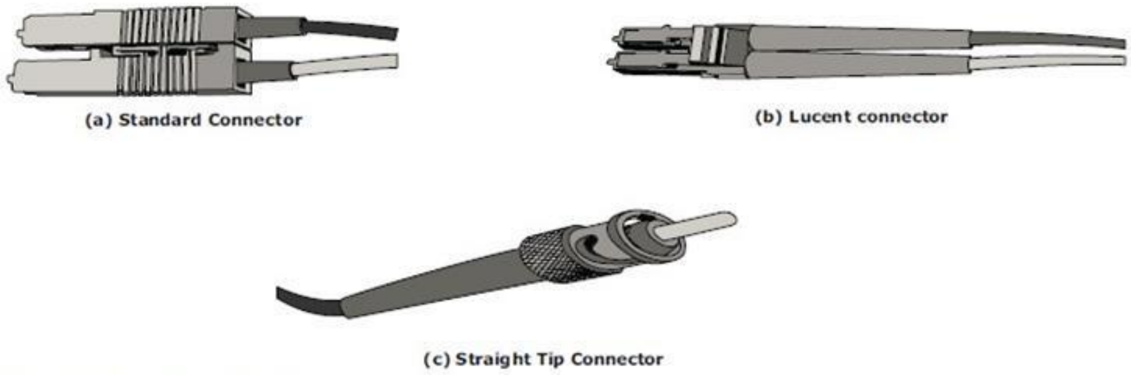


Fig 5-5: SC, LC, and ST connectors

5.3.3 Interconnect Devices

The commonly used interconnecting devices in SAN are

1) **Hubs,**
2) **Switches,**
3) **Directors**

- **Hubs** are used as communication devices in FC-AL implementations. Hubs physically connect nodes in a logical loop or a physical star topology.
- All the nodes must share the bandwidth because data travels through all the connection points. Because of availability of low cost and high performance switches, hubs are no longer used in SANs.
- **Switches** are more **intelligent** than hubs and directly **route data from one physical port to another**. Therefore, nodes do not share the bandwidth. Instead, each node has a dedicated communication path, resulting in bandwidth aggregation.
- Switches are available with:
  - ✓ Fixed port count
  - ✓ Modular design: port count is increased by installing additional port cards to open slots.
- **Directors are larger than switches** and are deployed for data center implementations.
- The function of directors is similar to that of FC switches, but directors have higher port count and fault tolerance capabilities.
- Port card or blade has multiple ports for connecting nodes and other FC switches

5.3.4 SAN Management Software

- SAN management software manages the interfaces between hosts, interconnect devices, and storage arrays.
- The software provides a view of the SAN environment and enables management of various resources from one central console.

- It provides key management functions, including mapping of storage devices, switches, and servers, monitoring and generating alerts for discovered devices, and logical partitioning of the SAN, called *zoning*