Course Outcomes

CO 1: Identify key challenges in managing information and analyze different storage networking technologies and virtualization

CO 2: Explain components and the implementation of NAS

CO 3: Describe CAS architecture and types of archives and forms of virtualization

CO 4: Illustrate the storage infrastructure and management activities

Institution Vision

To produce top-quality engineers who are groomed for attaining excellence in their profession and competitive enough to help in the growth of nation and global society.

Institution Mission

M1: To offer affordable high-quality graduate program in engineering with value education and make the students socially responsible.

M2: To support and enhance the institutional environment to attain research excellence in both faculty and students and to inspire them to push the boundaries of knowledge base.

M3: To identify the common areas of interest amongst the individuals for the effective industry- institute partnership in a sustainable way by systematically working together.

M4: To promote the entrepreneurial attitude and inculcate innovative ideas among the engineering professionals.

Department Vision

To be a center of excellence in Information Science & Engineering education, research and training to meet the growing needs of the industry and society.

Department Mission

M1: To impart theoretical and practical knowledge through the concepts and technologies in Information Science and Engineering

M2: To foster research, collaboration and higher education with premier institutions and industries.

M3: Promote innovation and entrepreneurship to fulfill the needs of the society and industry

Program Educational Objectives

PEO1: Analyse, design and implement solutions to the real-world problems in the field of Information Science and Engineering with multidisciplinary setup

PEO2: Keep abreast with the technology, innovation and pursue higher education with high standards of social and professional ethics

PEO3: Develop professional and entrepreneurship skills to work effectively as an individual and in a team to meet the ever-changing goals of the organization

Program Outcomes

PO1: **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: **Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3: **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

PO4: **Conduct investigations of complex problems:** Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: **Modern tool usage:** Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations

PO6: **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

PO7: **Environment sustainability:** Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions

PO11: **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to ones own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

Program Specific Outcome

PSO1: Design, implement and maintain the information systems that fulfill the current needs of the industry and society

PSO2: Apply computational theory, storage and networking concepts to solve the day to day problems of the world

RAID Implementation Methods

The two methods of RAID implementation are hardware and software. Both have their advantages and disadvantages.

Software RAID

Software ROAD uses host-based software to provide RAID functions. It is implemented at the operating-system level and does not use a dedicated hardware controller to manage the RAID array.

Software RAID implementations offer cost and simplicity benefits when compared with hardware RAID. However, they have the following limitations:

- Performance: Software RAID affects overall system performance. This is due to 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.

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

RAID Array Components

A RAID array is an enclosure that contains a number of disk drives and sup- porting 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 ROAD set or a RAID group.

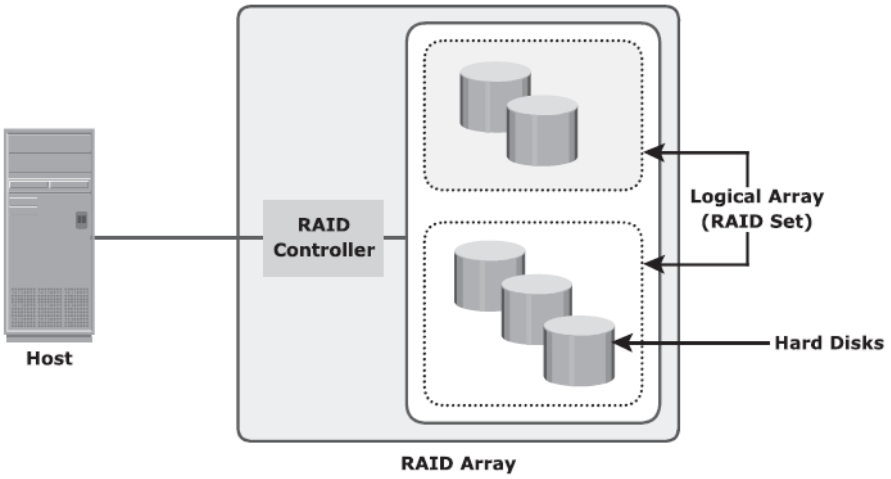


Figure: Components of a RAID array

RAID Techniques

RAID techniques — striping, mirroring, and parity — form the basis for defining various RAID levels. These techniques determine the data availability and performance characteristics of a RAID set.

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 address- able 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. Figure 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, assuming that the accessed data starts at the beginning of the strip. 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. For example, in a five disk striped RAID set with a strip size of 64 KB, the stripe size is 320 KB(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, as discussed in the following sections.

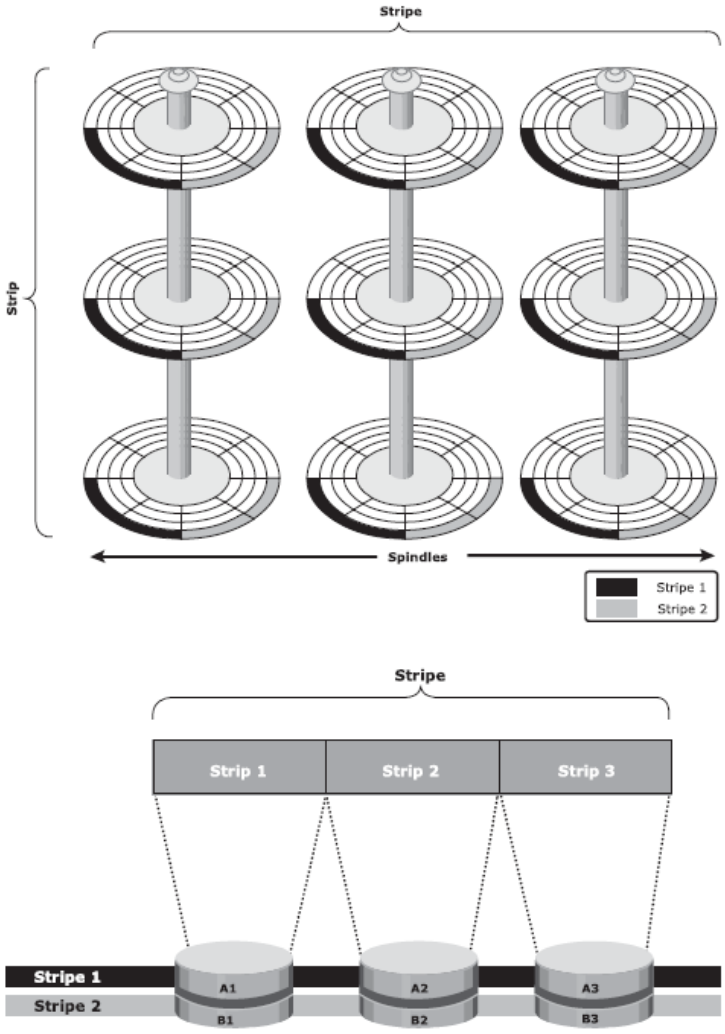


Figure: Striped RAID set

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 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. In addition to providing complete data redundancy, mirroring enables fast 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 the data.

Mirroring involves duplication of data —the amount of storage capacity needed is twice the amount of data being stored. Therefore, mirroring is considered expensive and is preferred for mission-critical applications that cannot afford the risk of any data loss. Mirroring improves read performance because read requests can be serviced by both disks. However, write performance is slightly lower than that in a single disk because each write request manifests as two writes on the disk drives. Mirroring does not deliver the same levels of write performance as a striped RAID.

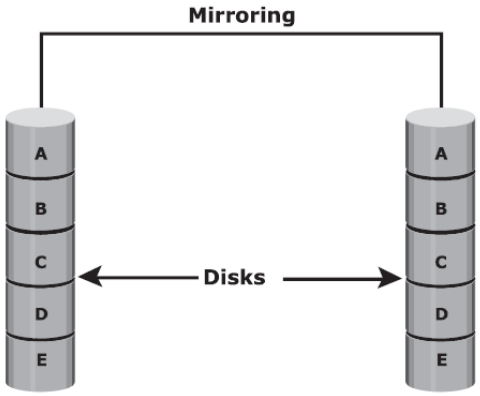


Figure: Mirrored disks in an array

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

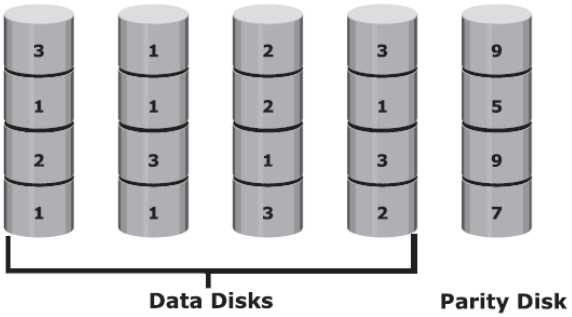


Figure 3-4: Parity

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 next. (A and B denote the inputs and C, the output after performing the XOR operation.) 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. For example, if a disk containing all the data from A fails, the data can be regenerated by performing an XOR between B and C.

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

Compared to mirroring, parity implementation considerably reduces the cost associated with data protection. Consider an example of a parity RAID configuration with five disks where four disks hold data, and the fifth holds the parity information. In this example, parity requires only 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 array.

For parity RAID, the stripe size calculation does not include the parity strip. For example 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).

RAID Levels

Application performance, data availability requirements, and cost determine the RAID level selection. These RAID levels are defined on the basis of striping, mirroring, and parity techniques. Some RAID levels use a single technique, whereas others use a combination of techniques. Table shows the commonly used RAID levels.

Table : Raid Levels

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

RAID 0

RAID 0 configuration uses data striping techniques, where data is striped across all the disks within a RAID set. Therefore it utilizes the full storage capacity of a RAID set. To read data, all the strips are put back together by the controller. Figure shows RAID 0 in an array in which data is striped across five disks. When the number of drives in the RAID set increases, performance improves because more data can be read or written simultaneously. RAID 0 is a good option for applications that need high I/O throughput.

However, if these applications require high availability during drive failures, RAID 0 does not provide data protection and availability.

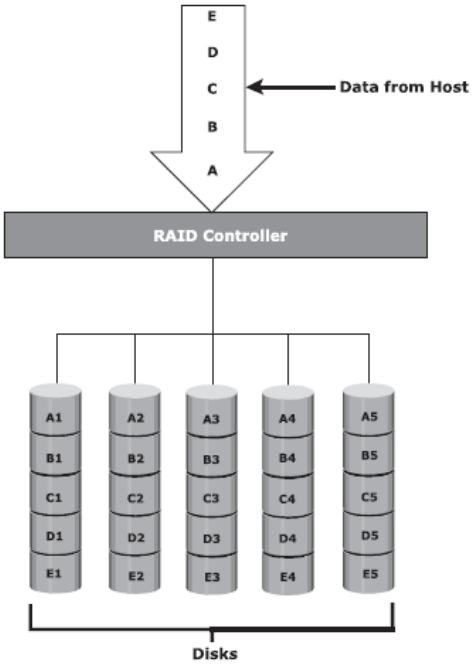


Figure: RAID 0

RAID 1

RAID 1 is based on the mirroring technique. In this RAID configuration, data is mirrored to provide fault tolerance. 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.

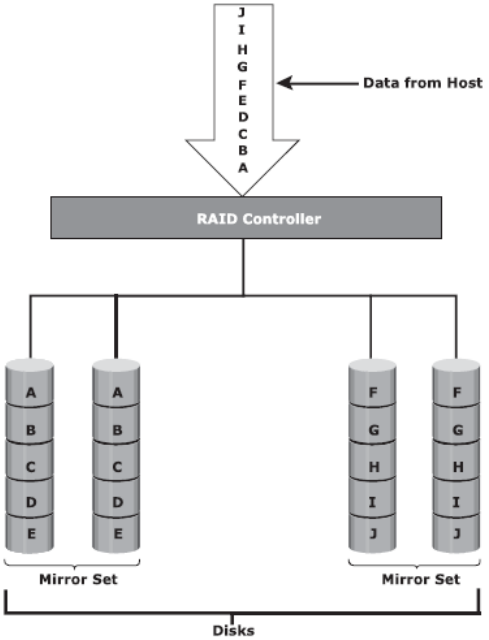
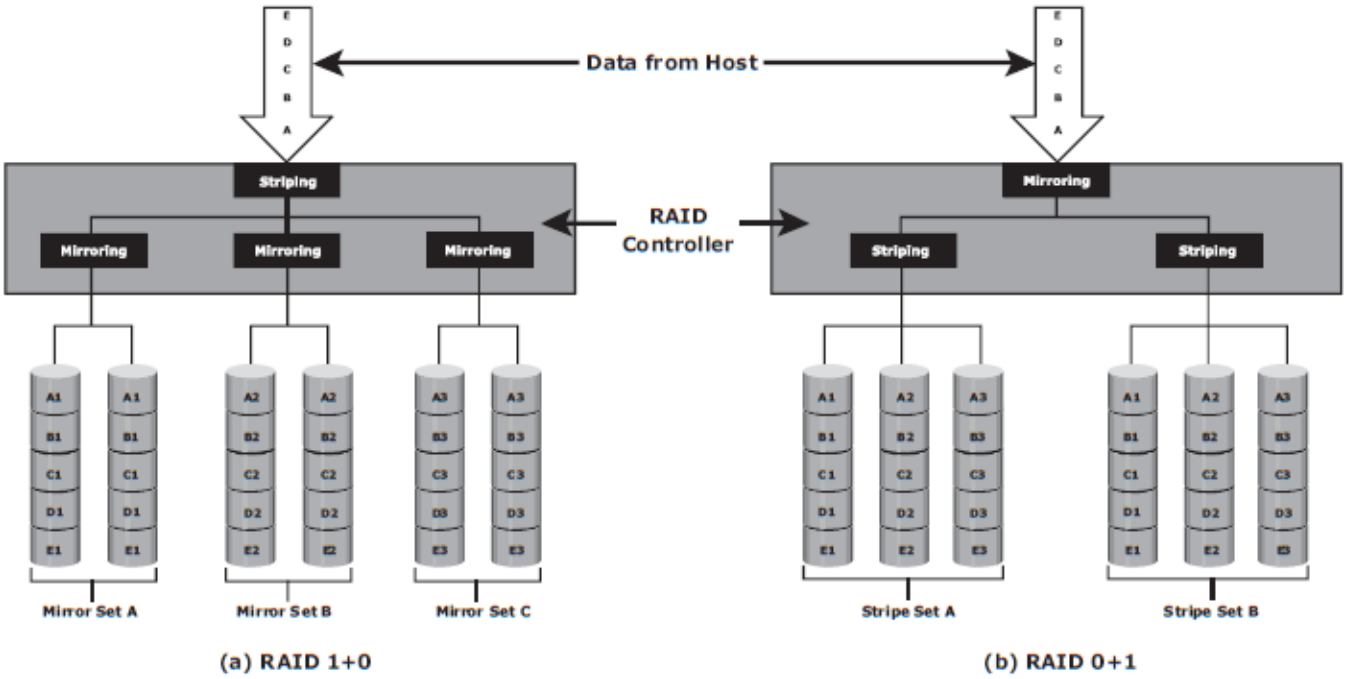


Figure: RAID 1

Nested RAID

Most data centers require data redundancy and performance from their RAID arrays. RAID 1+0 and RAID 0+1combine 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.



RAID 1+0 is also known as RAID 10 (Ten) or RAID 1/0. Similarly, RAID 0+1 is also known as RAID 01 or RAID 0/1. RAID 1+0 performs well for workloads 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

A common misconception is that RAID 1+0 and RAID 0+1 are the same. Under normal conditions, RAID levels 1+0 and 0+1 offer identical benefits. However, rebuild operations in the case of disk failure differ between the two.

RAID 1+0 is also called striped mirror. The basic element of RAID 1+0 is a mirrored pair, which means that data is first mirrored and then both copies of the data are striped across multiple disk drive pairs in a RAID set. When replacing a failed drive, only the mirror is rebuilt. In other words, the disk array controller uses the surviving drive in the mirrored pair for data recovery and continuous operation. Data from the surviving disk is copied to the replacement disk.

To understand the working of RAID 1+0, 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:

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. Consider the same example of six disks to understand the working of 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:

Drives 1 + 2 + 3 = RAID 0 (Stripe Set A)

Drives 4 + 5 + 6 = RAID 0 (Stripe Set B)

Now, 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.

RAID 3

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

RAID 3 provides good performance for applications that involve large sequen- tial data access, such as data backup or video streaming.

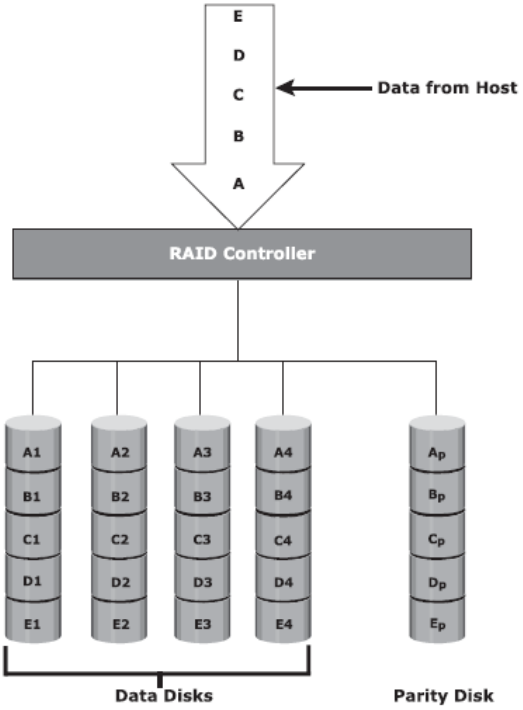


Figure: RAID 3

RAID 4

Similar to RAID 3, 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.

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

RAID 5

RAID 5 is a versatile RAID implementation. It is similar to RAID 4 because it uses striping. The drives (strips) are also independently accessible. The difference between RAID 4 and RAID 5 is the parity location. In RAID 4, parity is written to a dedicated drive, creating a write bottleneck for the parity disk. In RAID 5, parity is distributed across all disks to overcome the write bottleneck of a dedicated parity disk.

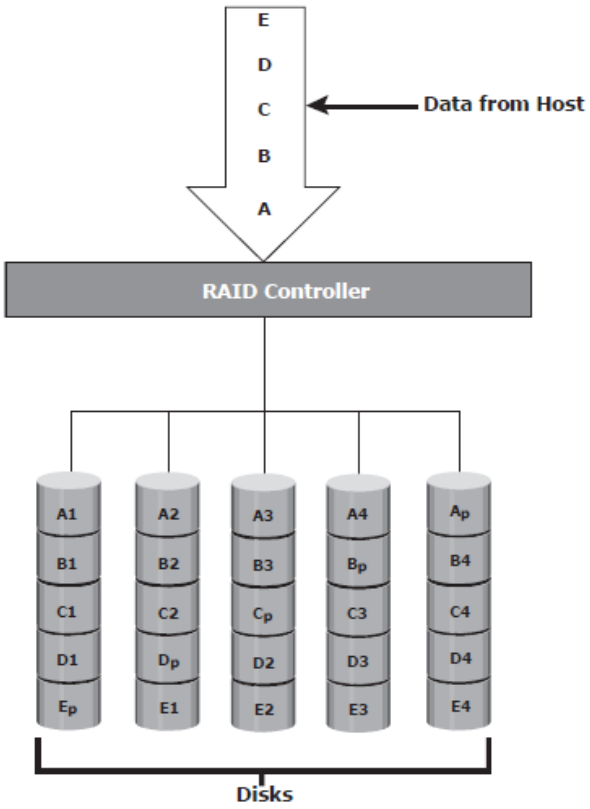


Figure: RAID 5

RAID 5 is good for random, read-intensive I/O applications and preferred for messaging, data mining, medium-performance media serving, and relational database management system (RDBMS) implementations, in which database administrators (DBAs) optimize data access.

RAID 6

RAID 6 works the same way as RAID 5, except that RAID 6 includes a second parity element to enable survival if two disk failures occur in a RAID set. Therefore, a RAID 6 implementation requires at least four disks. RAID 6 distributes the parity across all the disks. The write penalty (explained later in this chapter) in RAID 6 is more than that in RAID 5; therefore, RAID 5 writes perform better than RAID 6. The rebuild operation in RAID 6 may take longer than that in RAID 5 due to the presence of two parity sets.

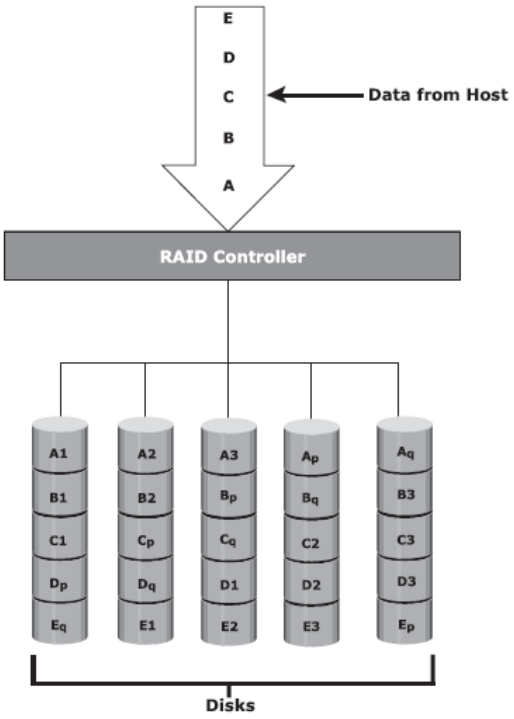


Figure: RAID 6

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 and parity RAID configurations, every write operation translates into more I/O overhead for the disks, which is referred to as a write penalty. In a RAID 1 implementation, every write operation must be performed on two disks configured as a mirrored pair, whereas in a RAID 5 implementation, a write operation may manifest as four I/O operations. When performing I/Os to a disk configured with RAID 5, the controller has to read, recalculate, and write a parity segment for every data write operation.

Figure illustrates a single write operation on RAID 5 that contains a group of five disks.

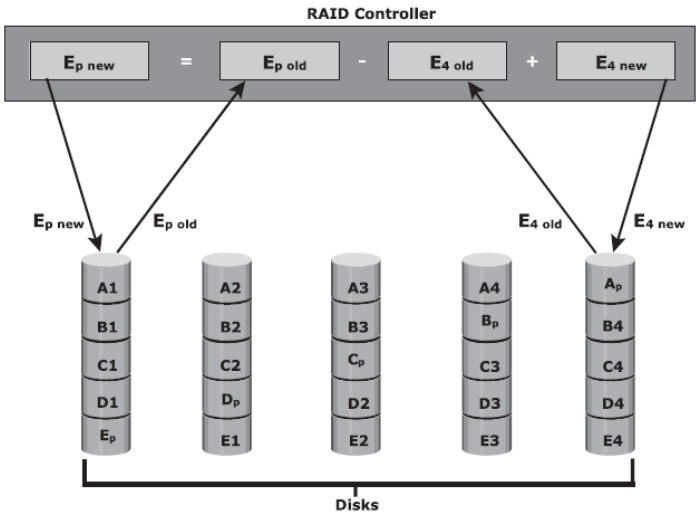


Figure: Write penalty in RAID 5

The parity (P) at the controller is calculated as follows:

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

Whenever the controller performs a write I/O, parity must be computed by reading the old parity (Ep old) and the old data (E4 ,d) from the disk, which means two read I/Os. Then, the new parity (Ep new) is computed as follows:

Ep new = Ep old - E_4 old + E4 new (XOR operations)

After computing the new parity, the controller completes the write I/O by writing the new data and the new parity onto the disks, amounting to two write I/Os. Therefore, the controller performs two disk reads and two disk writes for every write operation, and the write penalty 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.

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 x 5,200 + 4 x (0.4 x 5,200) [because the write penalty for RAID 5 is 4]

= 3,120 + 4 x 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 x 5,200 + 2 x (0.4 x 5,200) [because every write manifests as two writes to the disks]

= 3,120 + 2 x 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)

RAID Comparison

Table 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

Intelligent Storage Systems

Business-critical applications require high levels of performance, availability, security, and scalability. A 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 disk drives and their mechanical components. RAID technology made an important contribution to enhancing storage performance and reliability, but disk drives, even with a RAID implementation, could not meet the performance requirements of today's applications.

With advancements in technology, a new breed of storage solutions, known as *intelligent storage systems*, has evolved. These 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. These arrays have an operating environment that intelligently and optimally handles the management, allocation, and utilization of storage resources. Support for flash drives and other modern-day technologies, such as virtual storage provisioning and automated storage tiering, has added a new dimension to storage system performance, scalability, and availability.

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 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 the 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*).

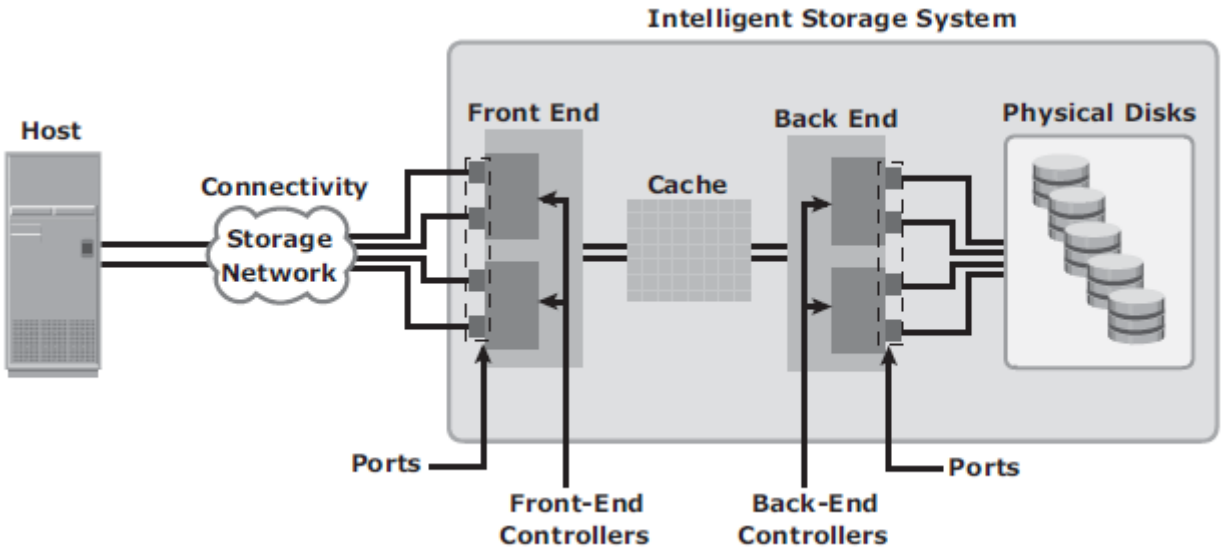


Figure 4-1: Components of an intelligent storage system

Front End

The front end provides the interface between the storage system and the host. It consists of two components: front-end ports and front-end controllers. Typically, a front end has redundant controllers for high availability, and each controller contains multiple 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.

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 and in the disk.

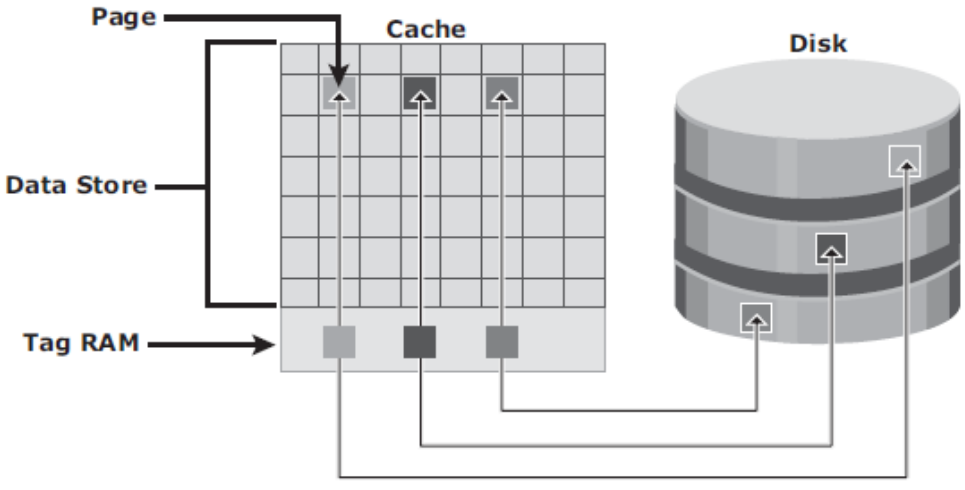


Figure 4-2: Structure of cache

Entries in tag RAM indicate where data is found in cache and where the data belongs on the disk. Tag RAM includes a *dirty bit* flag, which indicates whether the data in cache has been committed to the disk. It also contains time-based information, such as the time of last access, which is used to identify cached information that has not been accessed for a long period and may be freed up.

Read Operation with Cache

When a host issues a read request, the storage controller reads the tag RAM to determine whether the required data is available in cache. If the requested data is found in the cache, it is called a *read cache hit* or *read hit* and data is sent directly to the host, without any disk operation. This provides a fast response time to the host (about a millisecond). If the requested data is not found in cache, it is called a *cache miss* and the data must be read from the disk. The back end accesses the appropriate disk and retrieves the requested data. Data is then placed in cache and finally sent to the host through the front end. Cache misses increase the I/O response time.

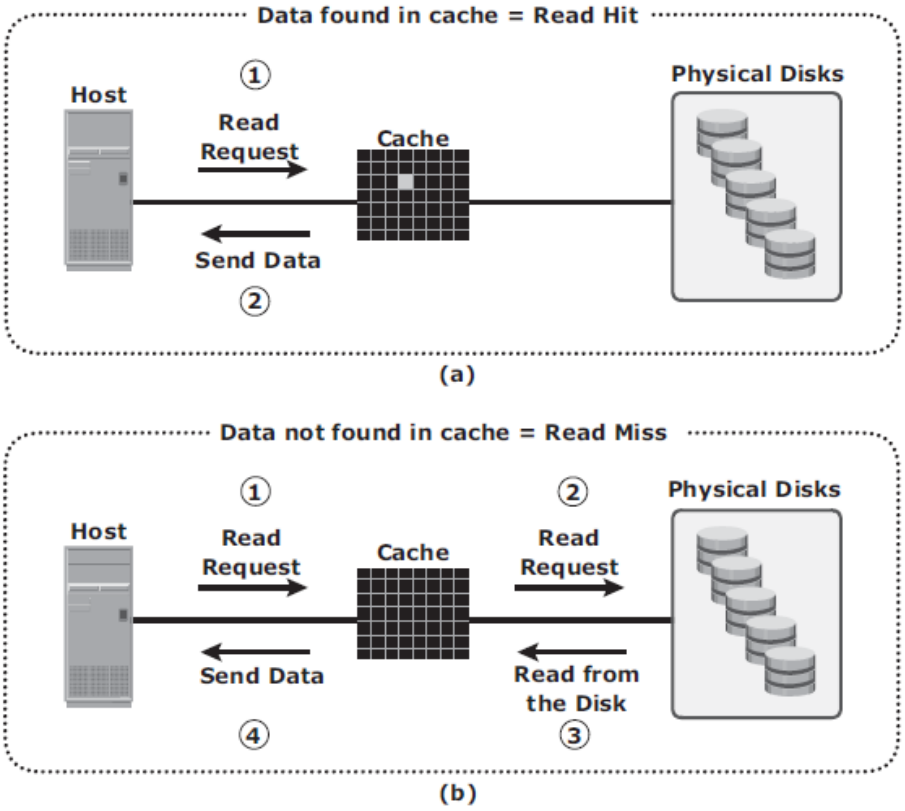


Figure 4-3: Read hit and read miss

A *prefetch* 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 prefetch*, the intelligent storage system prefetches a fixed amount of data. It is most suitable when host I/O sizes are uniform. In *variable prefetch*, the storage system prefetches an amount of data in multiples of the size of the host request. *Maximum prefetch* limits the number of data blocks that can be prefetched to prevent the disks from being rendered busy with prefetch at the expense of other I/Os.

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.

Write Operation with Cache

Write operations with cache provide performance advantages over writing directly to disks. When an I/O is written to cache and acknowledged, it is completed in far less time (from the host's perspective) than it would take to write directly to disk. Sequential writes also offer opportunities for optimization because many smaller writes can be coalesced for larger transfers to disk drives with the use of cache.

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

- **Write-back cache:** Data is placed in cache and an acknowledgment is sent to the host immediately. Later, data from several writes are committed (de-staged) to the disk. Write response times are much faster because the write operations are isolated from the mechanical delays of the disk. However, uncommitted data is at risk of loss if cache failures occur.
- **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 the 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 particularly 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. Typically, the read cache is small, but it should be increased if the application being used is read-intensive. In other global cache implementations, the ratio of cache available for reads versus writes is dynamically adjusted based on the workloads.

Cache Management

Cache is a finite and expensive resource that needs proper management. Even though 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 discussed in the following list:

- **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 that has not been accessed for a while will not be requested by the host. However, if a page contains write data that has not yet been committed to disk, the data is first written to disk before the page is reused.
- **Most Recently Used (MRU):** This algorithm is the opposite of LRU, where 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 Figure 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.

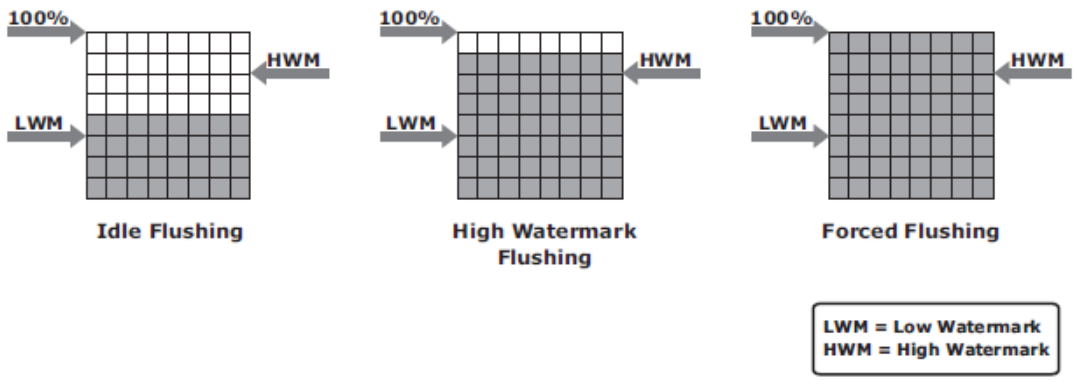


Figure 4-4: Types of flushing

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* and *cache vaulting*:

- **Cache mirroring:** Each write to cache is held in two different memory locations on two independent memory cards. If a cache failure occurs, 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, if a cache failure occurs, the data can still be accessed from the disk. Because only writes are mirrored, this method results in better utilization of the available cache. In cache mirroring approaches, the problem of maintaining *cache coherency* is introduced. Cache coherency means that data in two different cache locations must be identical at all times. It is the responsibility of the array operating environment to ensure coherency.
- **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 or 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. Therefore, storage vendors use a set of physical disks to dump the contents of cache during power failure. This is called *cache vaulting* and the disks are called *vault drives*. When power is restored, data from these disks is written back to write cache and then written to the intended disks.

Server flash-caching technology uses intelligent caching software and a PCI Express (PCIe) flash card on the host. This dramatically improves application performance by reducing latency, and accelerates throughput. Server flash caching technology works in both physical and virtual environments and provides performance acceleration for read-intensive workloads. This technology uses minimal CPU and memory resources from the server by offloading flash management onto the PCIe card. It intelligently determines which data would benefit by sitting in the server on PCIe flash and closer to the application. This avoids the latencies associated with I/O access over the network to the storage array. With this, the processing power required for an application's most frequently referenced data is offloaded from the back-end storage to the PCIe card. Therefore, the storage array can allocate greater processing power to other applications.

Back End

The *back end* provides an interface between cache and the physical disks. It consists of two components: back-end ports and back-end controllers. The back-end controls data transfers between cache and the physical disks. From cache, data is sent to the back end and then routed to the destination disk.

Physical disks are connected to ports on the back end. The back-end controller communicates with the disks when performing reads and writes and also provides additional, but limited, temporary data storage. The algorithms implemented on back-end controllers provide error detection and correction, along with RAID functionality. For high data protection and high availability, storage systems are configured with dual controllers with multiple ports. Such configurations provide an alternative path to physical disks if a controller or port failure occurs. This reliability is further enhanced if the disks are also dual-ported. In that case, each disk port can connect to a separate controller. Multiple controllers also facilitate load balancing.

Physical Disk

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

Storage Provisioning

Storage provisioning is the process of assigning storage resources to hosts based on capacity, availability, and performance requirements of applications running on the hosts. Storage provisioning can be performed in two ways: traditional and virtual.

Virtual provisioning leverages virtualization technology for provisioning storage for applications.

Traditional Storage Provisioning

In traditional storage provisioning, physical disks are logically grouped together and a required RAID level is applied to form a set, called a RAID set. The number of drives in the RAID set and the RAID level determine the availability, capacity, and performance of the RAID set. It is highly recommend that the RAID set be created from drives of the same type, speed, and capacity to ensure maximum usable capacity, reliability, and consistency in performance. For example, if drives of different capacities are mixed in a RAID set, the capacity of the smallest drive is used from each disk in the set to make up the RAID set's overall capacity. The remaining capacity of the larger drives remains unused. Likewise, mixing higher revolutions per minute (RPM) drives with lower RPM drives lowers the overall performance of the RAID set.

RAID sets usually have a large capacity because they combine the total capacity of individual drives in the set. Logical units are created from the RAID sets by partitioning (seen as slices of the RAID set) the available capacity into smaller units.

These units are then assigned to the host based on their storage requirements. Logical units are spread across all the physical disks that belong to that set. Each logical unit created from the RAID set is assigned a unique ID, called a logical unit number (LUN). LUNs hide the organization and composition of the RAID set from the hosts. LUNs created by traditional storage provisioning methods are also referred to as thick LUNs to distinguish them from the LUNs created by virtual provisioning methods.

Figure 4-5 shows a RAID set consisting of five disks that have been sliced, or partitioned, into two LUNs: LUN 0 and LUN 1. These LUNs are then assigned to Host1 and Host 2 for their storage requirements.

When a LUN is configured and assigned to a non-virtualized host, a bus scan is required to identify the LUN. This LUN appears as a raw disk to the operating system. To make this disk usable, it is formatted with a fi le system and then the file system is mounted.

In a virtualized host environment, the LUN is assigned to the hypervisor, which recognizes it as a raw disk. This disk is configured with the hypervisor file system, and then virtual disks are created on it. Virtual disks are fi les on the hypervisor file system. The virtual disks are then assigned to virtual machines and appear as raw disks to them. To make the virtual disk usable to the virtual machine, similar steps are followed as in a non-virtualized environment. Here, the LUN space may be shared and accessed simultaneously by multiple virtual machines.

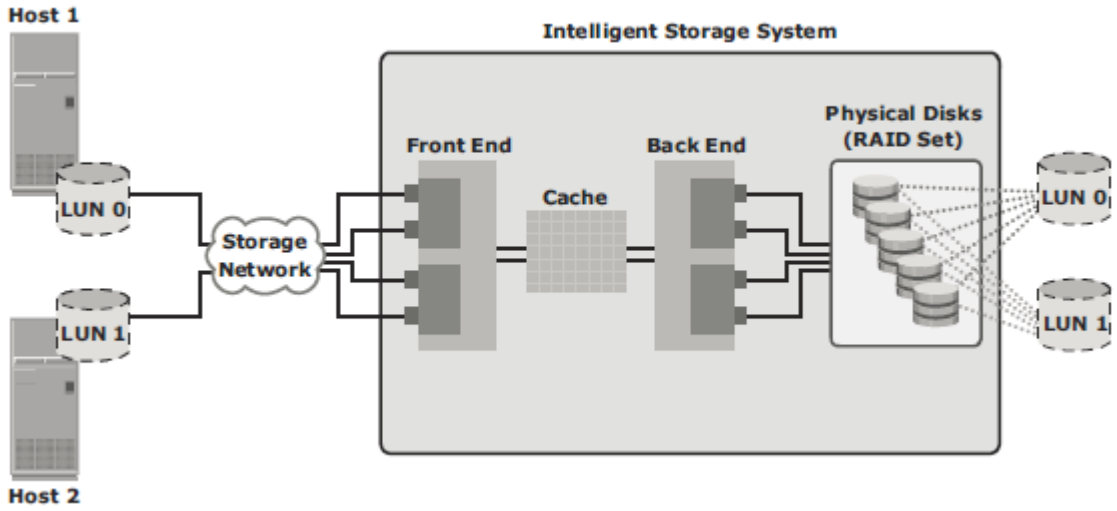


Figure 4-5: RAID set and LUNs

Virtual machines can also access a LUN directly on the storage system. In this method the entire LUN is allocated to a single virtual machine. Storing data in this way is recommended when the applications running on the virtual machine are response-time sensitive, and sharing storage with other virtual machines may impact their response time. The direct access method is also used when a virtual machine is clustered with a physical machine. In this case, the virtual machine is required to access the LUN that is being accessed by the physical machine.

LUN Expansion: MetaLUN

MetaLUN is a method to expand LUNs that require additional capacity or performance. A metaLUN can be created by combining two or more LUNs. A metaLUN consists of a base LUN and one or more component LUNs. MetaLUNs can be either concatenated or striped.

Concatenated expansion simply adds additional capacity to the base LUN. In this expansion, the component LUNs are not required to be of the same capacity as the base LUN. All LUNs in a concatenated metaLUN must be either protected (parity or mirrored) or unprotected (RAID 0). RAID types within a metaLUN can be mixed. For example, a RAID 1/0 LUN can be concatenated with a RAID 5 LUN.

However, a RAID 0 LUN can be concatenated only with another RAID 0 LUN. Concatenated expansion is quick but does not provide any performance benefit. (See Figure 4-6.)

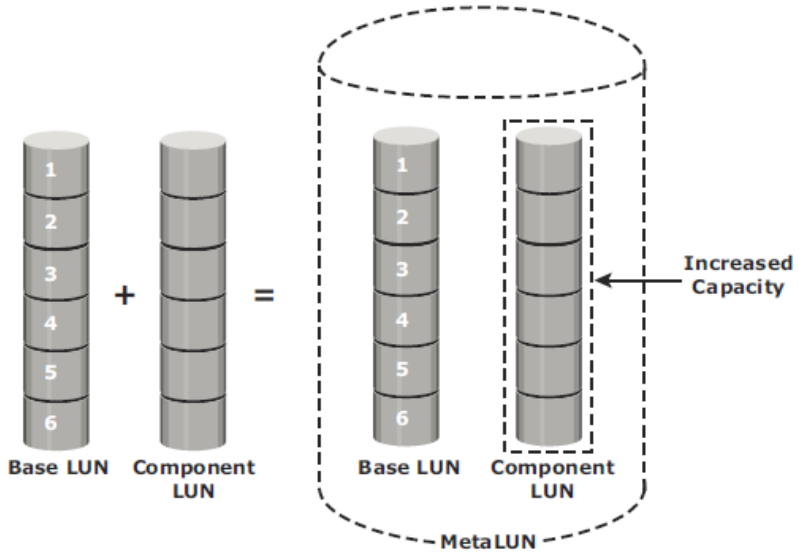


Figure 4-6: Concatenated metaLUN

Striped expansion restripes the base LUN's data across the base LUN and component LUNs. In striped expansion, all LUNs must be of the same capacity and RAID level. Striped expansion provides improved performance due to the increased number of drives being striped (see Figure 4-7).

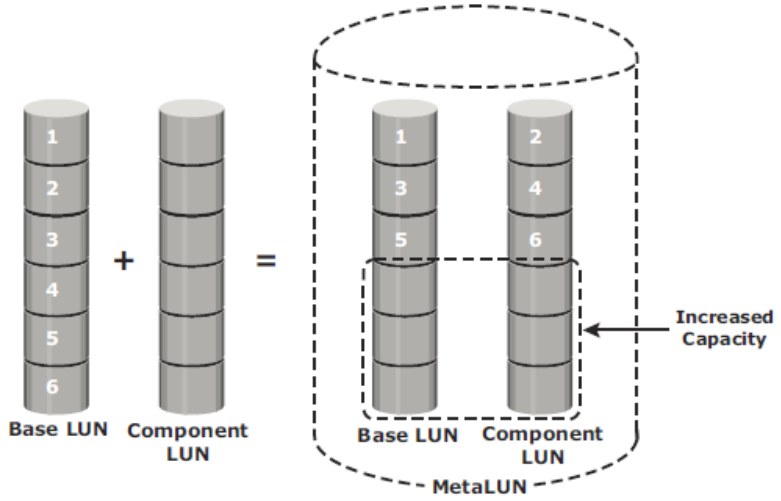


Figure 4-7: Striped metaLUN

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

Virtual Storage Provisioning

Virtual provisioning enables creating and presenting a LUN with more capacity than is physically allocated to it on the storage array. The LUN created using virtual provisioning is called a thin LUN to distinguish it from the traditional LUN.

Thin LUNs do not require physical storage to be completely allocated to them at the time they are created and presented to a host. Physical storage is allocated to the host "on-demand" from a shared pool of physical capacity. A shared pool consists of physical disks. A shared pool in virtual provisioning is analogous to a RAID group, which is a collection of drives on which LUNs are created. Similar to a RAID group, a shared pool supports a single RAID protection level. However, unlike a RAID group, a shared pool might contain large numbers of drives. Shared pools can be homogeneous (containing a single drive type) or heterogeneous (containing mixed drive types, such as flash, FC, SAS, and SATA drives).

Virtual provisioning enables more efficient allocation of storage to hosts. Virtual provisioning also enables oversubscription, where more capacity is presented to the hosts than is actually available on the storage array. Both shared pool and thin LUN can be expanded nondisruptively as the storage requirements of the hosts grow. Multiple shared pools can be created within a storage array, and a shared pool may be shared by multiple thin LUNs. Figure 4-8 illustrates the provisioning of thin LUNs.

Comparison between Virtual and Traditional Storage Provisioning

Administrators typically allocate storage capacity based on anticipated storage requirements. This generally results in the over provisioning of storage capacity, which then leads to higher costs and lower capacity utilization. Administrators often over-provision storage to an application for various reasons, such as, to avoid frequent provisioning of storage if the LUN capacity is exhausted, and to reduce disruption to application availability. Over provisioning of storage often leads to additional storage acquisition and operational costs.

Virtual provisioning addresses these challenges. Virtual provisioning improves storage capacity utilization and simplifies storage management. Figure 4-9 shows an example, comparing virtual provisioning with traditional storage provisioning.

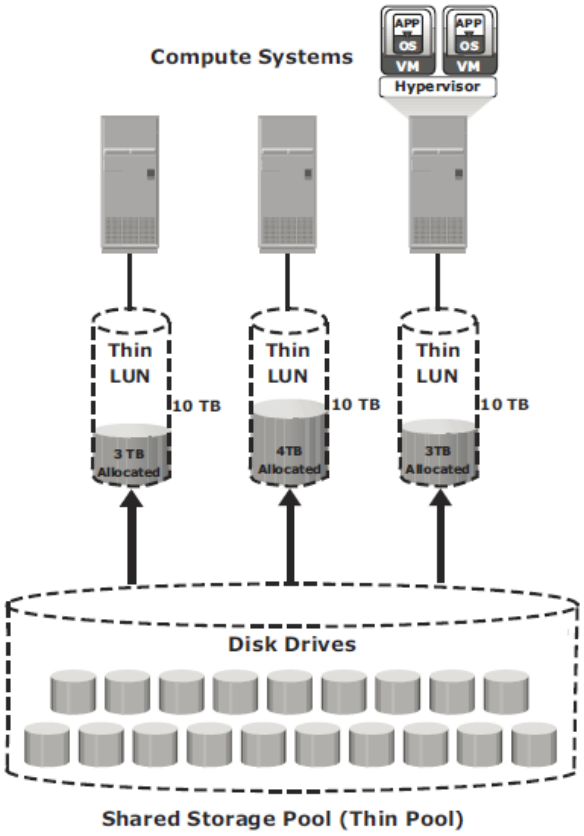


Figure 4-8: Virtual provisioning

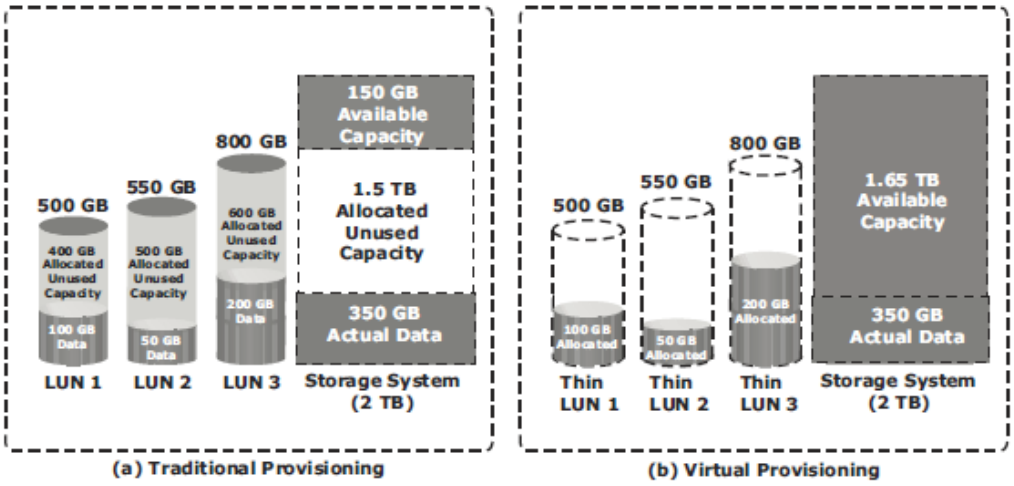


Figure 4-9: Traditional versus virtual provisioning

With traditional provisioning, three LUNs are created and presented to one or more hosts (see Figure 4-9 [a]). The total storage capacity of the storage system is 2 TB. The allocated capacity of LUN 1 is 500 GB, of which only 100 GB is consumed, and the remaining 400 GB is unused. The size of LUN 2 is 550 GB, of which 50 GB is consumed, and 500 GB is unused. The size of LUN 3 is 800 GB, of which 200 GB is consumed, and 600 GB is unused. In total, the storage system has 350 GB of data, 1.5 TB of allocated but unused capacity, and only 150 GB of remaining capacity available for other applications.

Now consider the same 2 TB storage system with virtual provisioning (see Figure 4-9 [b]). Here, three thin LUNs of the same sizes are created. However, there is no allocated unused capacity. In total, the storage system with virtual provisioning has the same 350 GB of data, but 1.65 TB of capacity is available for other

applications, whereas only 150 GB is available in traditional storage provisioning. Use Cases for Thin and Traditional LUNs Virtual provisioning and thin LUN offer many benefits, although in some cases traditional LUN is better suited for an application. Thin LUNs are appropriate for applications that can tolerate performance variations. In some cases, performance improvement is perceived when using a thin LUN, due to striping across a large number of drives in the pool. However, when multiple thin LUNs contend for shared storage resources in a given pool, and when utilization reaches higher levels, the performance can degrade. Thin LUNs provide the best storage space efficiency and are suitable for applications where space consumption is difficult to forecast. Using thin LUNs benefits organizations in reducing power and acquisition costs and in simplifying their storage management.

Traditional LUNs are suited for applications that require predictable performance. Traditional LUNs provide full control for precise data placement and allow an administrator to create LUNs on different RAID groups if there is any workload contention. Organizations that are not highly concerned about storage space efficiency may still use traditional LUNs.

Both traditional and thin LUNs can coexist in the same storage array. Based on the requirement, an administrator may migrate data between thin and traditional LUNs.

LUN Masking

LUN masking is a process that provides data access control by defining which LUNs a host can access. The LUN masking function is implemented on the storage array. This ensures that volume access by hosts is controlled appropriately, preventing unauthorized or accidental use in a shared environment.

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

Types of Intelligent Storage Systems

Intelligent storage systems generally fall into one of the following two categories:

- High-end storage systems
- Midrange storage systems

Traditionally, 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.

High-End Storage Systems

High-end storage systems, referred to as *active-active arrays*, are generally aimed at large enterprise applications. These systems 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 through any of the available controllers.

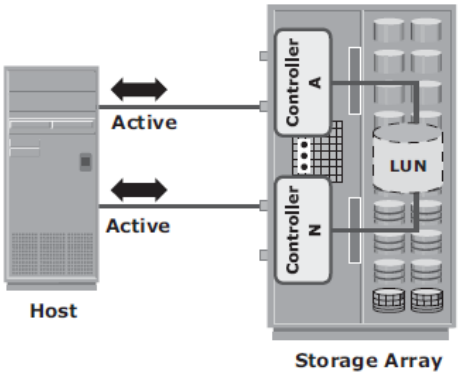


Figure 4-10: Active-active configuration

To address 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 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 hosts and applications
- Support for array-based local and remote data replication

In addition to these features, high-end systems possess some unique features that are required for mission-critical applications.

Midrange Storage Systems

Midrange storage systems are also referred to as *active-passive arrays* and are best suited for small- and medium-sized enterprise applications. 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 controller that owns the LUN. As shown in Figure 4-11, the host can perform reads or writes to the LUN only through the path to controller A because 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 interface to disk drives.

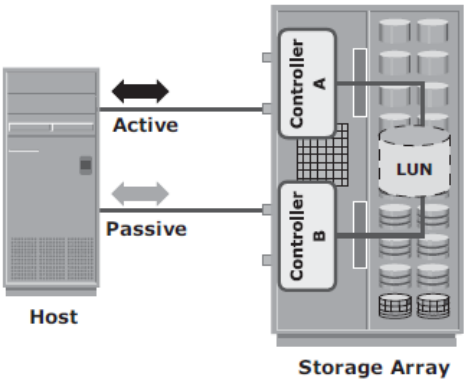


Figure 4-11: Active-passive configuration

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. However, they ensure high redundancy and high performance for applications with predictable workloads. They also support array-based local and remote replication.

Fibre Channel Storage Area Networks

Organizations are experiencing an explosive growth in information. This information needs to be stored, protected, optimized, and managed efficiently. Data center managers are burdened with the challenging task of providing low-cost, high-performance information management solutions. An effective information management solution must provide the following:

- **Just-in-time information to business users:** Information must be available to business users when they need it. 24 x 7 data availability is becoming one of the key requirements of today's storage infrastructure. The explosive growth in storage, proliferation of new servers and applications, and the spread of mission-critical data throughout enterprises are some of the challenges that need to be addressed to provide information availability in real time.
- **Integration of information infrastructure with business processes:** The storage infrastructure should be integrated with various business processes without compromising its security and integrity.
- **Flexible and resilient storage infrastructure:** The storage infrastructure must provide flexibility and resilience that aligns with changing business requirements. Storage should scale without compromising the performance requirements of applications and, at the same time, the total cost of managing information must be low.

Direct-attached storage (DAS) is often referred to as a stovepiped storage environment. Hosts "own" the storage, and it is difficult to manage and share resources on these isolated storage devices. Efforts to organize this dispersed data led to the emergence of the *storage area network* (SAN). SAN is a high-speed, dedicated network of servers and shared storage devices. A SAN provides storage consolidation and facilitates centralized data management. It meets the storage demands efficiently with better economies of scale and also provides effective maintenance and protection of data. Virtualized SAN and block storage virtualization provide enhanced utilization and collaboration among dispersed storage resources. The implementation of virtualization in SAN provides improved productivity, resource utilization, and manageability.

Common SAN deployments are Fibre Channel (FC) SAN and IP SAN. Fibre Channel SAN uses Fibre Channel protocol for the transport of data, commands, and status information between servers (or hosts) and storage devices. IP SAN uses IP-based protocols for communication.

Fibre Channel: Overview

The FC architecture forms the fundamental construct of the FC SAN infrastructure. *Fibre Channel* is a high-speed network technology that runs on high-speed optical fiber cables and serial copper cables. The FC technology was developed to meet the demand for increased speeds of data transfer between servers and mass storage systems. Although FC networking was introduced in 1988, the FC standardization process

began when the American National Standards Institute (ANSI) chartered the Fibre Channel Working Group (FCWG). By 1994, the new high-speed computer interconnection standard was developed and the Fibre Channel Association (FCA) was founded with 70 charter member companies. Technical Committee T11, which is the committee within International Committee for Information Technology Standards (INCITS), is responsible for Fibre Channel interface standards.

High data transmission speed is an important feature of the FC networking technology. The initial implementation offered a throughput of 200 MB/s (equivalent to a raw bit rate of 1Gb/s), which was greater than the speeds of Ultra SCSI (20 MB/s), commonly used in DAS environments. In comparison with Ultra SCSI, FC is a significant leap in storage networking technology. The latest FC implementations of 16 GFC (Fibre Channel) offer a throughput of 3200 MB/s (raw bit rates of 16 Gb/s), whereas Ultra640 SCSI is available with a throughput of 640 MB/s. The FC architecture is highly scalable, and theoretically, a single FC network can accommodate approximately 15 million devices.

The SAN and Its Evolution

A SAN carries data between servers (or *hosts*) and storage devices through Fibre Channel network. 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.

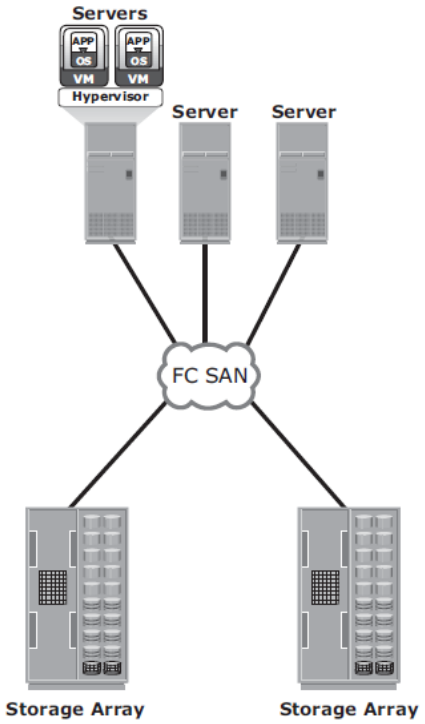


Figure 5-1: FC SAN implementation

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. This enhanced data accessibility to applications across the enterprise. Now, FC-AL has been almost abandoned for FC SANs due to its limitations but still survives as a back-end connectivity option to disk drives. Figure 5-2 illustrates the FC SAN evolution from FC-AL to enterprise SANs.

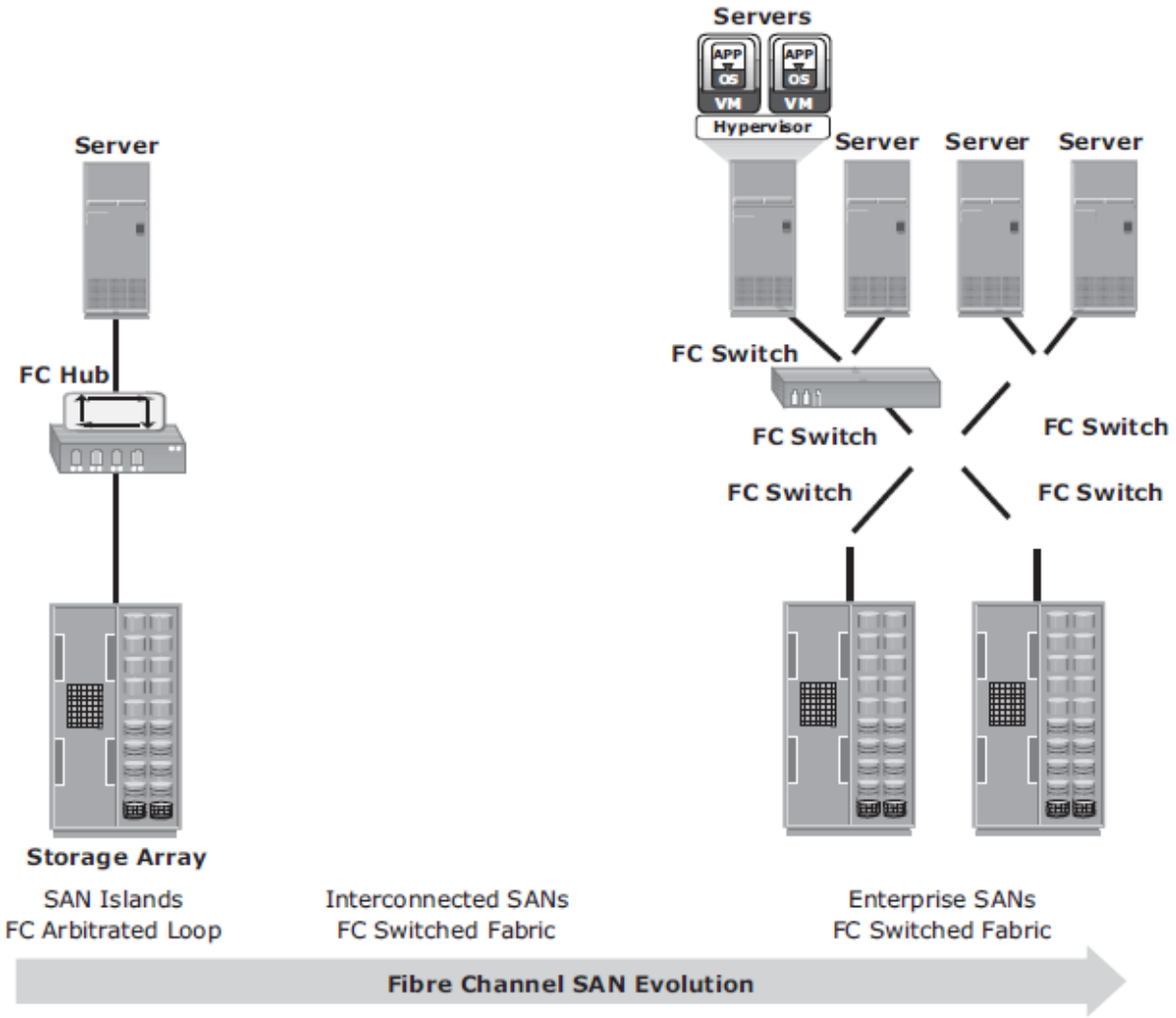


Figure 5-2: FC SAN evolution

Node Ports

In a Fibre Channel network, the end devices, such as hosts, storage arrays, and tape libraries, are all referred to as *nodes*. Each node is a source or destination of information. Each node requires one or more ports to provide a physical interface for communicating with other nodes. These ports are integral components of host adapters, such as HBA, and storage front-end controllers or adapters. In an FC environment a port operates in full-duplex data transmission mode with a *transmit* (Tx) link and a *receive* (Rx).

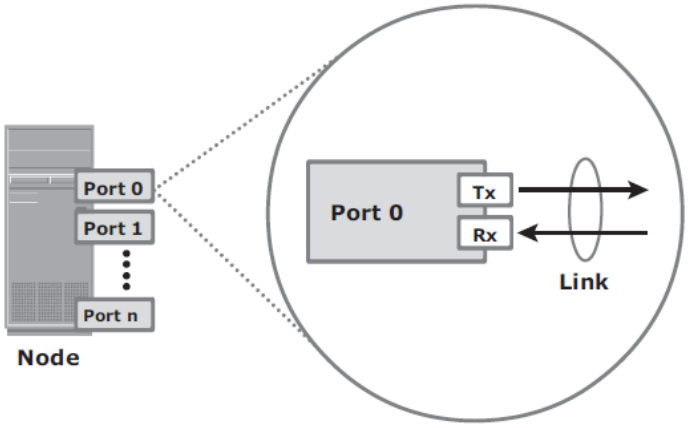


Figure 5-3: Nodes, ports, and links

Cables and Connectors

SAN implementations use optical fiber cabling. Copper can be used for shorter distances for back-end connectivity because it provides an acceptable signal-to noise ratio for distances up to 30 meters. Optical fiber cables carry data in the form of light. There are two types of optical cables: multimode and single-mode.

Multimode fiber (MMF) cable carries multiple beams of light projected at different angles simultaneously onto the core of the cable. Based on the bandwidth, multimode fibers are classified as OM1 (62.5µm core), OM2 (50µm core), and laser-optimized OM3 (50µm core). 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*. An MMF cable is typically used for short distances because of signal degradation (attenuation) due to modal dispersion.

Single-mode fiber (SMF) carries a single ray of light projected at the center of the core. These cables are available in core diameters of 7 to 11 microns; the most common size is 9 microns. 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 help to limit modal dispersion. Among all types of fiber cables, singlemode provides minimum signal attenuation over maximum distance (up to 10 km). A single-mode cable is used for long-distance cable runs, and distance usually depends on the power of the laser at the transmitter and sensitivity of the receiver.

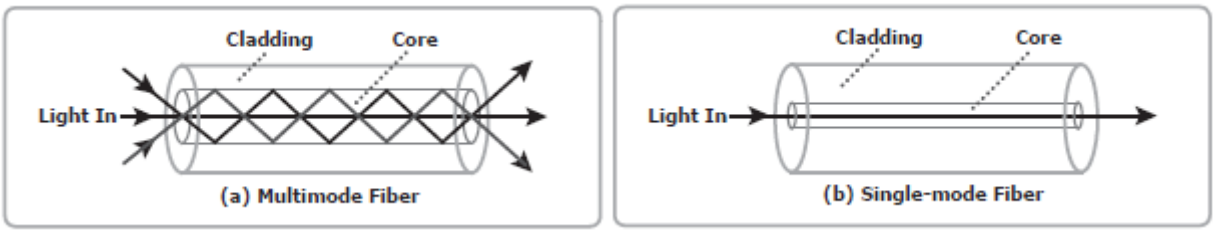


Figure 5-4: Multimode fiber and single-mode fiber

MMFs are generally used within data centers for shorter distance runs, whereas SMFs are used for longer distances. A connector is attached at the end of a cable to enable swift connection and disconnection of the cable to and from a port. A *Standard connector* (SC) and a *Lucent connector* (LC) are two commonly used

connectors for fiber optic cables. *Straight Tip* (ST) is another fiber-optic connector, which is often used with fiber patch panels.

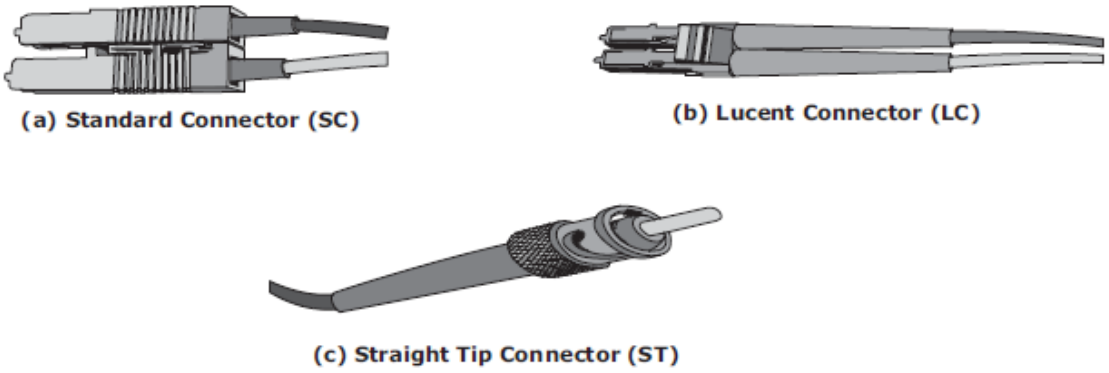


Figure 5-5: SC, LC, and ST connectors

Interconnect Devices

FC hubs, switches, and directors are the interconnect devices commonly used in FC SAN.

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 loop because data travels through all the connection points. Because of the availability of low-cost and high-performance switches, hubs are no longer used in FC 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.

Directors are high-end switches with a higher port count and better fault tolerance capabilities.

Switches are available with a fixed port count or with modular design. In a modular switch, the port count is increased by installing additional port cards to open slots. The architecture of a director is always modular, and its port count is increased by inserting additional line cards or blades to the director's chassis. High-end switches and directors contain redundant components to provide high availability. Both switches and directors have management ports (Ethernet or serial) for connectivity to SAN management servers.

A port card or blade has multiple ports for connecting nodes and other FC switches. Typically, a Fibre Channel transceiver is installed at each port slot that houses the transmit (Tx) and receive (Rx) link. In a transceiver, the Tx and Rx links share common circuitry. Transceivers inside a port card are connected to an application specifi c integrated circuit, also called port ASIC. Blades in a director usually have more than one ASIC for higher throughput.

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 *zoning.*

Scalability and performance are the primary differences between switches and hubs. Addressing in a switched fabric supports more than 15 million nodes within the fabric, whereas the FC-AL implemented in hubs supports only a maximum of 126 nodes.

Fabric switches provide full bandwidth between multiple pairs of ports in a fabric, resulting in a scalable architecture that supports multiple simultaneous communications.

Hubs support only one communication at a time. They provide a low-cost connectivity expansion solution. Switches, conversely, can be used to build dynamic, high-performance fabrics through which multiple communications can take place simultaneously. Switches are more expensive than hubs.

Question Bank

1. What is RAID? Explain the RAID levels RAID 0, 1 and Nested.
2. Define RAID. Explain RAID levels 3, 4, 5, 6.
3. With neat diagram explain the structure of read and write operations with cache.
4. Explain Fibre Channel connectivity with FC SAN implementation.
5. Describe with neat diagram the components of Intelligent Storage System.
6. Differentiate between Software and Hardware RAID.
7. Discuss how parity method is used in increasing fault tolerance in RAID levels.
8. Compare virtual and traditional storage provisioning.
9. What are RAID implementation methods?