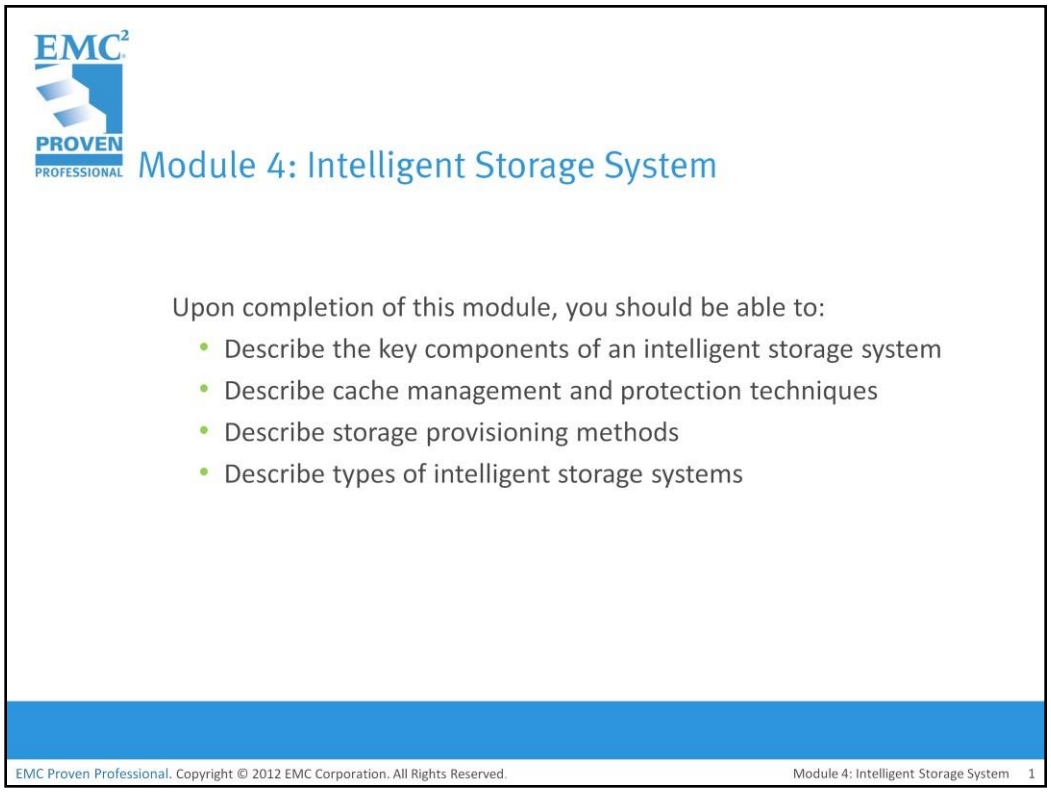
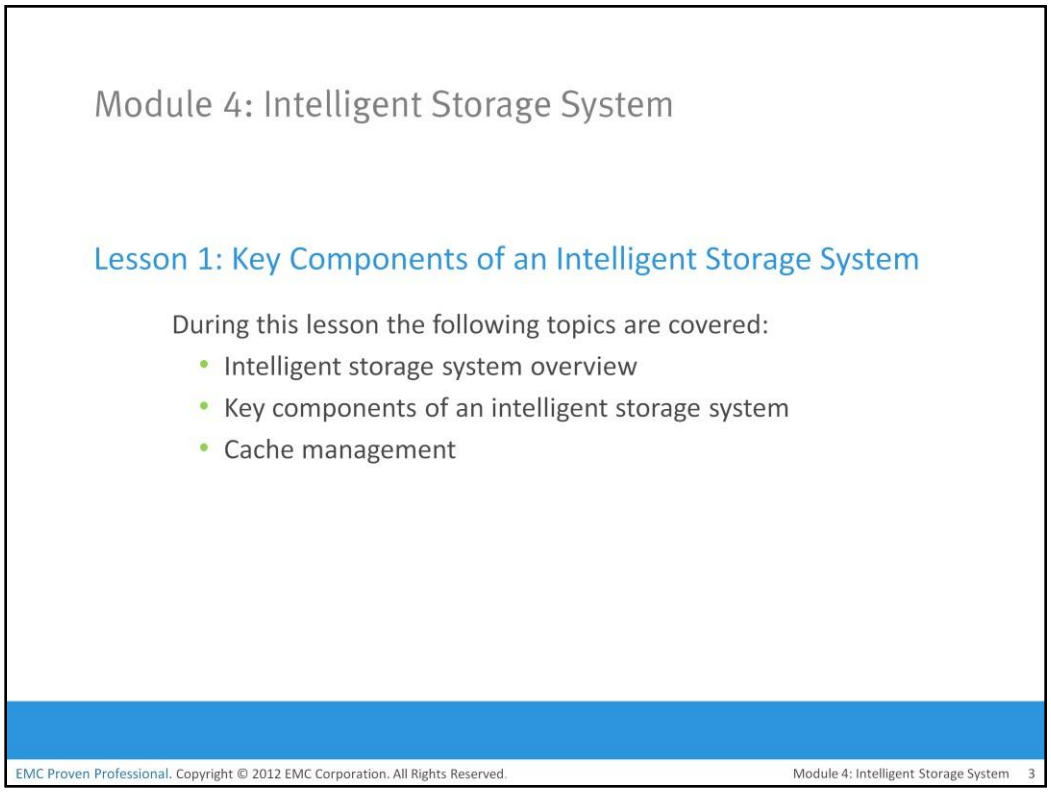
Module – 4

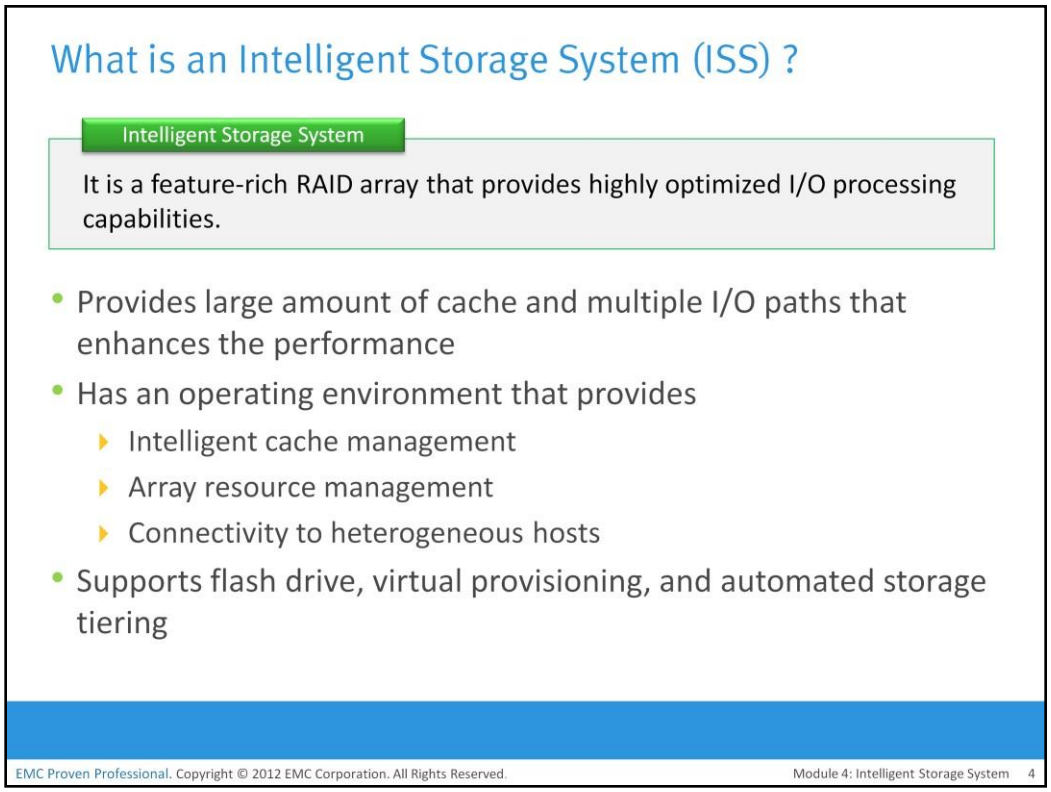
Intelligent Storage System



This module focuses on the key components of an intelligent storage system. It details the function of each component, including cache management and protection techniques. The module also focuses on the two storage provisioning methods. Finally, it describes the two types of intelligent storage systems.

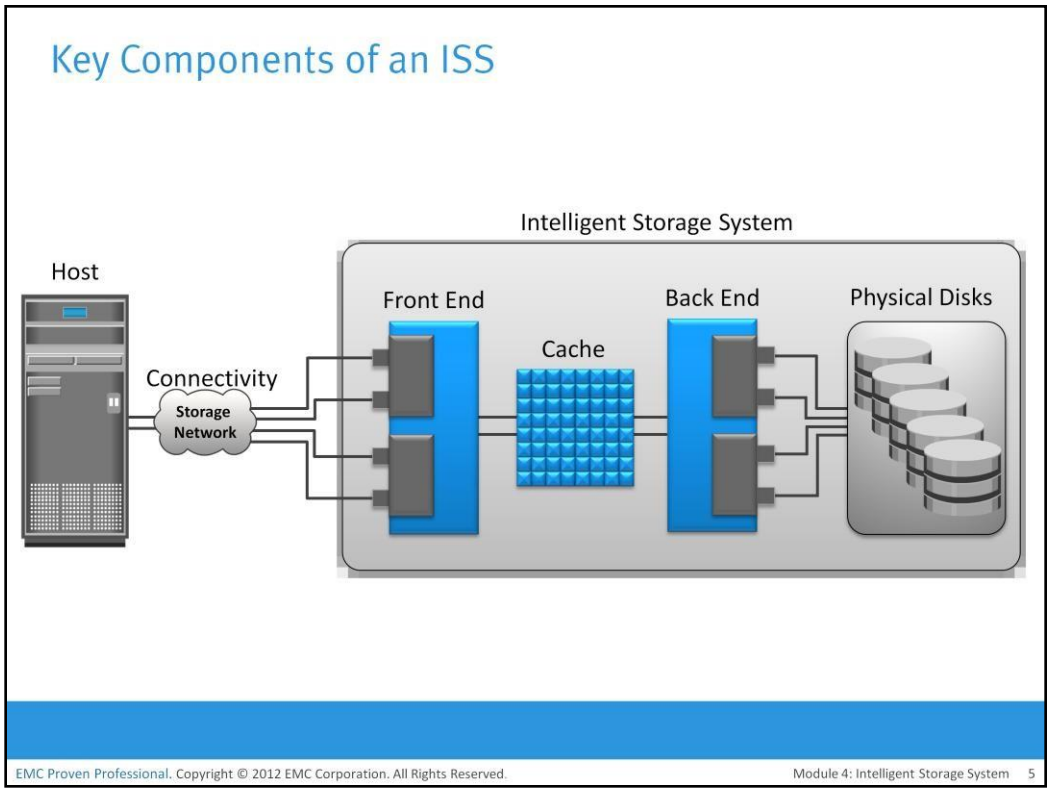


This lesson focuses on intelligent storage system overview and key components of an intelligent storage system. This lesson also focuses on cache management.

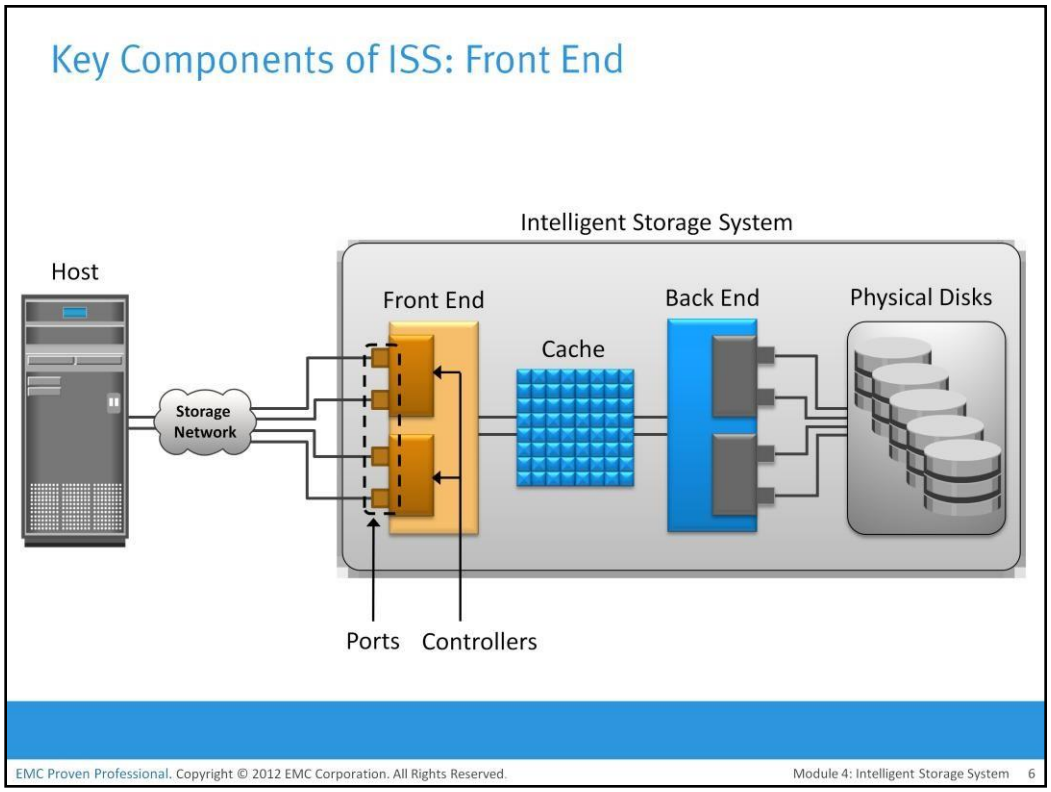


Business-critical applications require high levels of performance, availability, security, and scalability. A disk drive is a core element of storage that governs the performance of any storage system. Some of the older disk-array technologies could not overcome performance constraints due to the limitations 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.

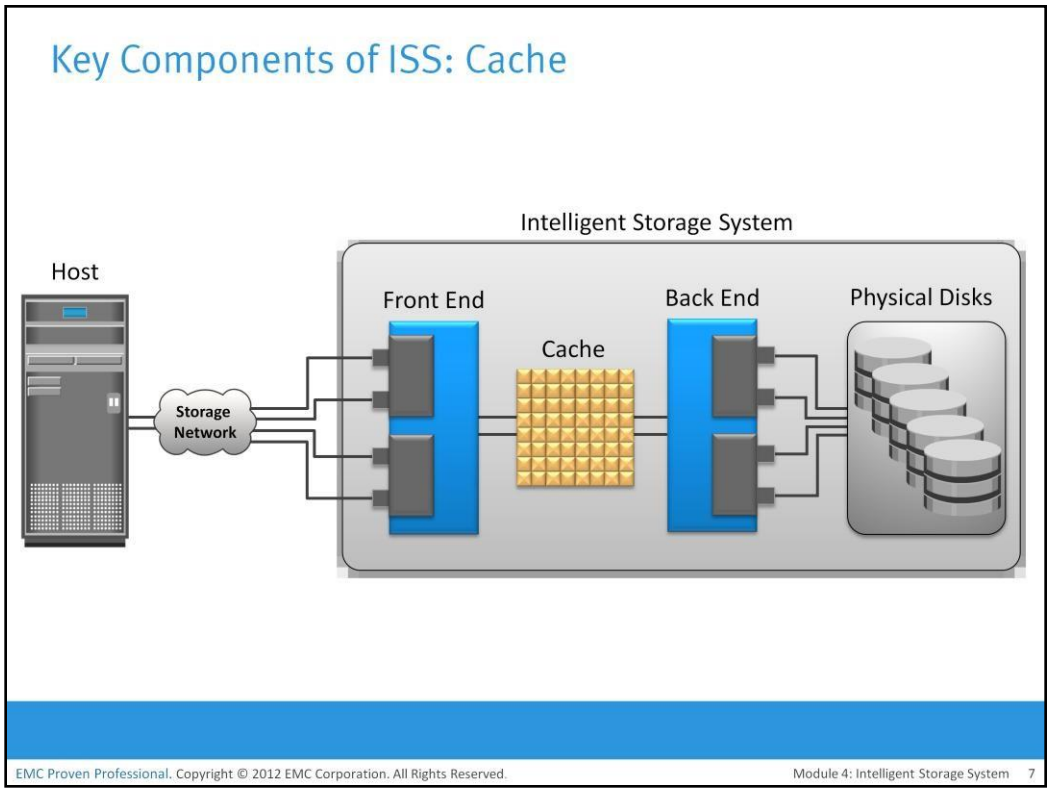
With advancements in technology, a new breed of storage solutions, known as *intelligent storage systems*, has evolved. These intelligent storage systems are feature-rich RAID arrays that provide highly optimized I/O processing capabilities. These storage systems are configured with a large amount of 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.



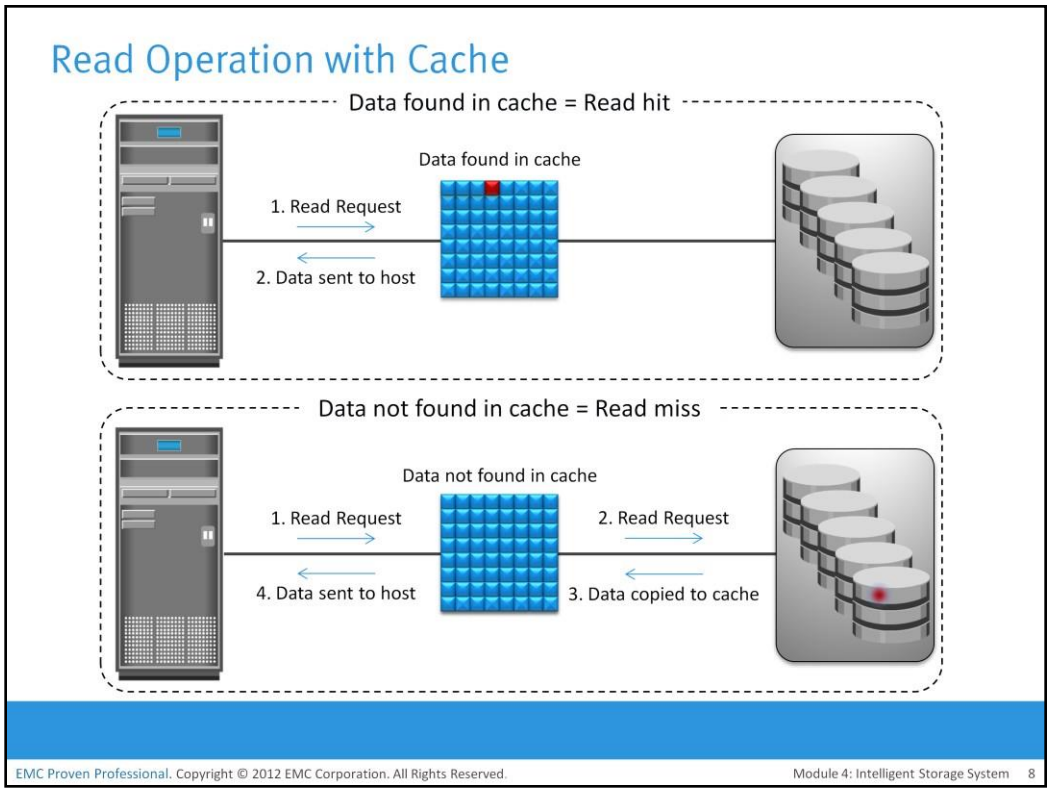
An intelligent storage system consists of four key components: *front end*, *cache*, *back end*, and *physical disks*. An I/O request received from the host at the front-end port is processed through cache and back end, to enable storage and retrieval of data from the physical disk. A read request can 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 as a *storage processor* or *storage controller)*.



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.



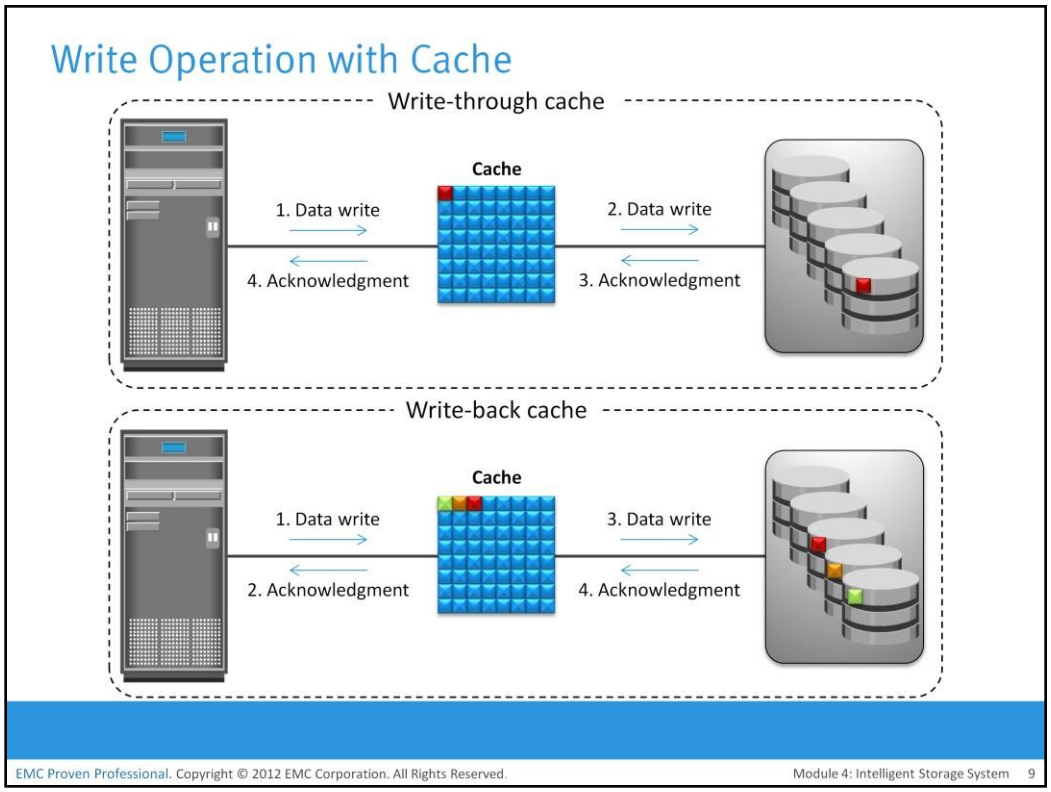
Cache is semiconductor memory where data is placed temporarily to reduce the time required to service I/O requests from the host. Cache improves storage system performance by isolating hosts from the mechanical delays associated with rotating disks or hard disk drive (HDD). Rotating disks are the slowest 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.



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.

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 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:

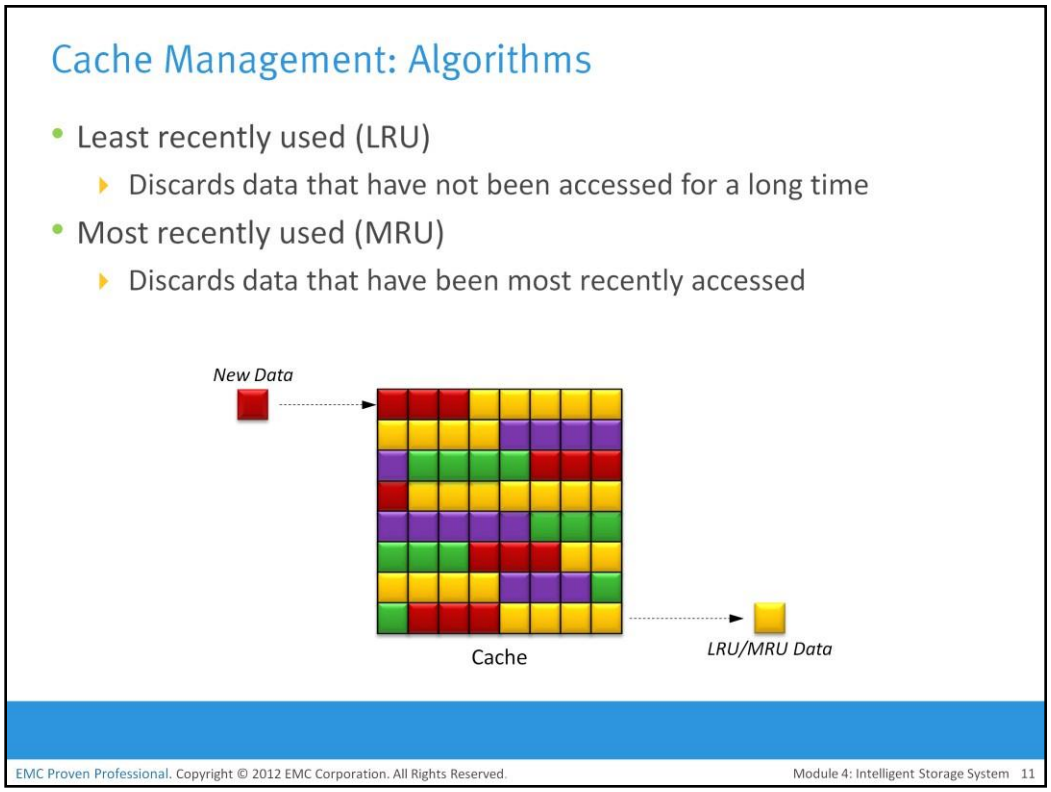
- Write-back cache: Data is placed in cache and an acknowledgment is sent to the host immediately. Later, data from several writes are committed (de-staged) to the disk. Write response times are much faster because the write operations are isolated from the mechanical delays of the disk. However, uncommitted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

Cont..

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.

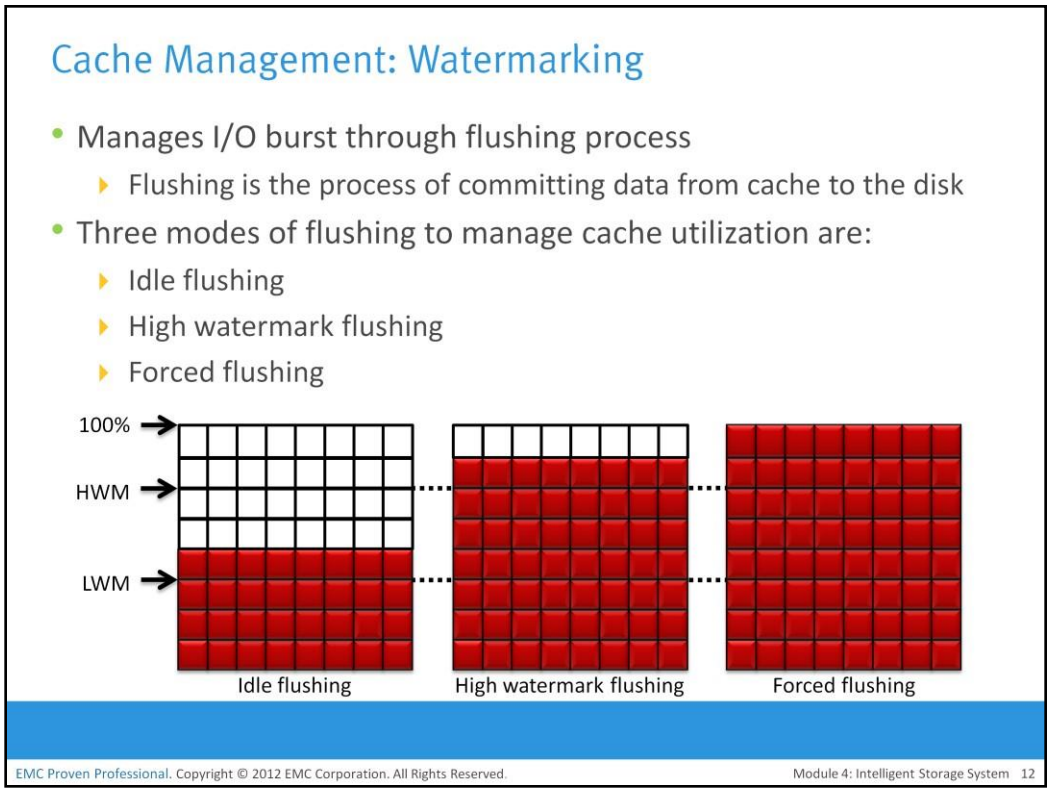
Global cache allows users to specify the percentages of cache available for reads and writes for cache management. Typically, the read cache is small, but it should be increased if the application being used is read-intensive. In other global cache implementations, the ratio of cache available for reads versus writes is dynamically adjusted based on the workloads.



Cache 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.

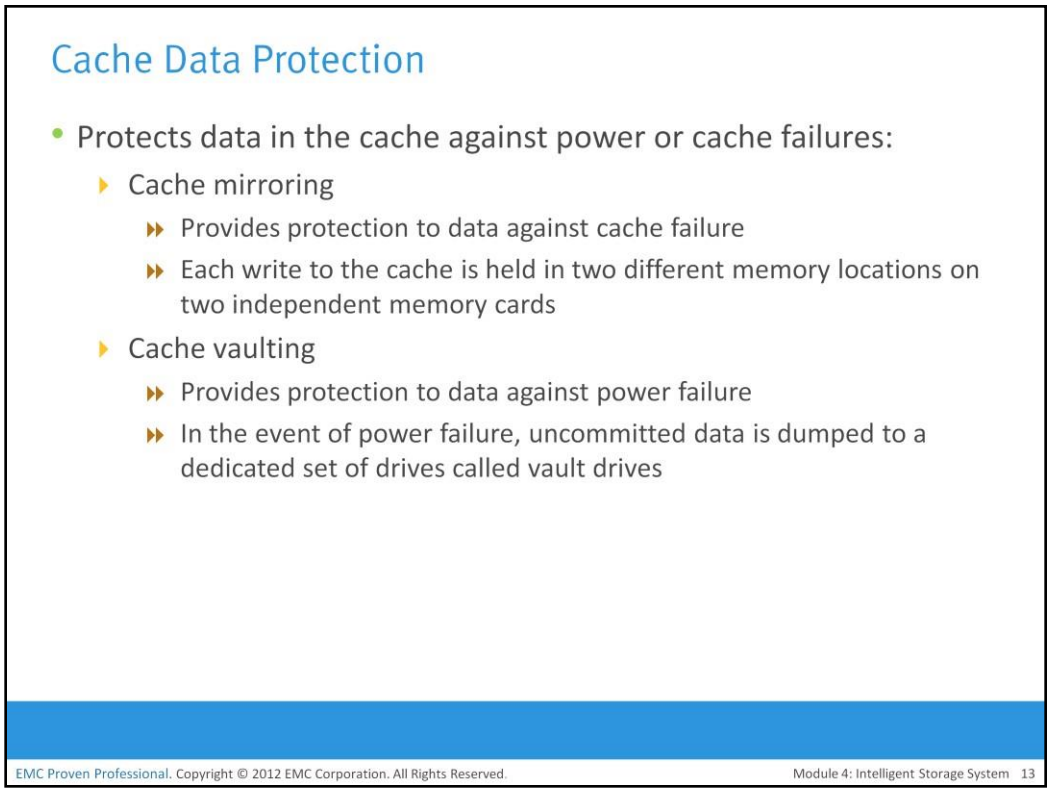
The most commonly used algorithms are discussed in the following list:

- 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.



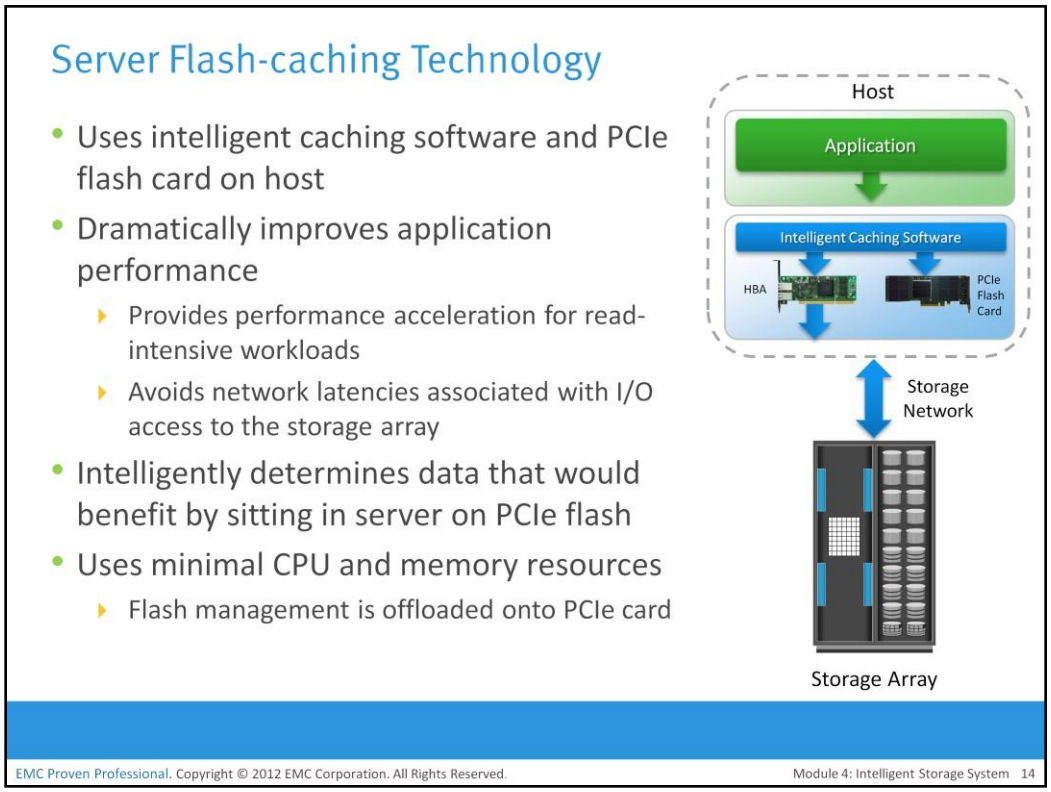
As cache fills, the storage system must take action to flush dirty pages (data written into the cache but not yet written to the disk) to manage space availability. *Flushing* is the process that commits data from cache to the disk. On the basis of the I/O access rate and pattern, high and low levels 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 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.



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.

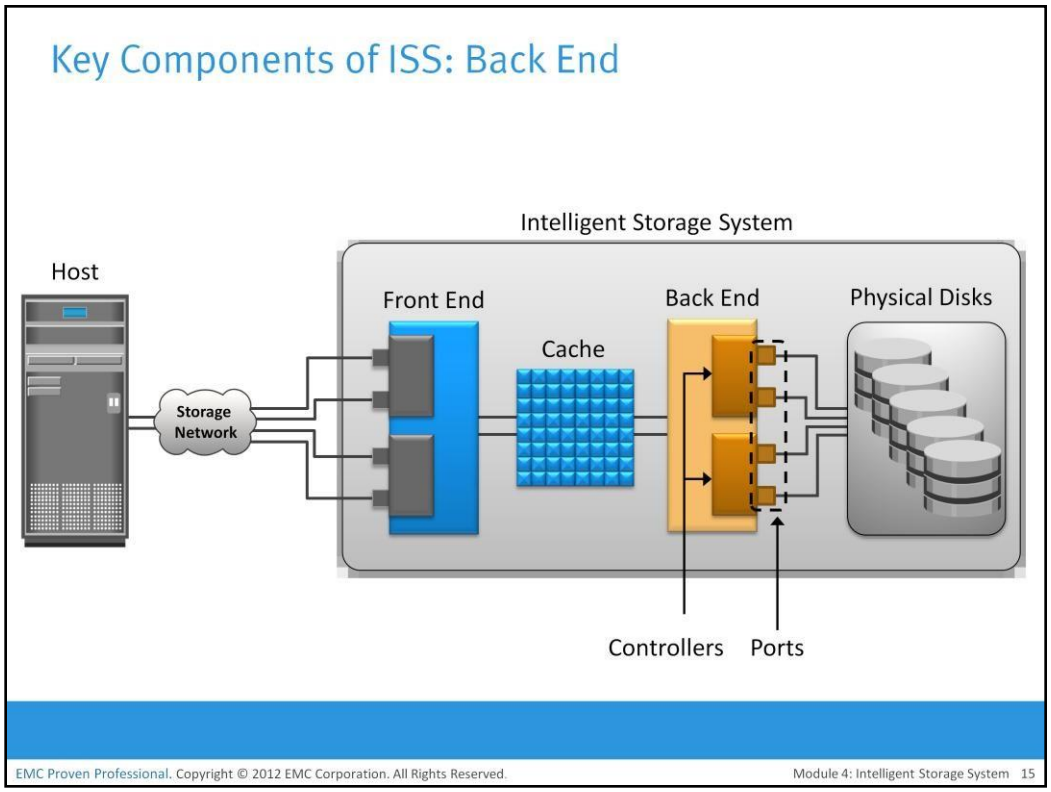


Server flash-caching technology uses intelligent caching software and PCI Express (PCIe) flash card on the host. This dramatically improves application performance by reducing latency and accelerates throughput. Server flash-caching technology works in both physical and virtual environments and provides 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.

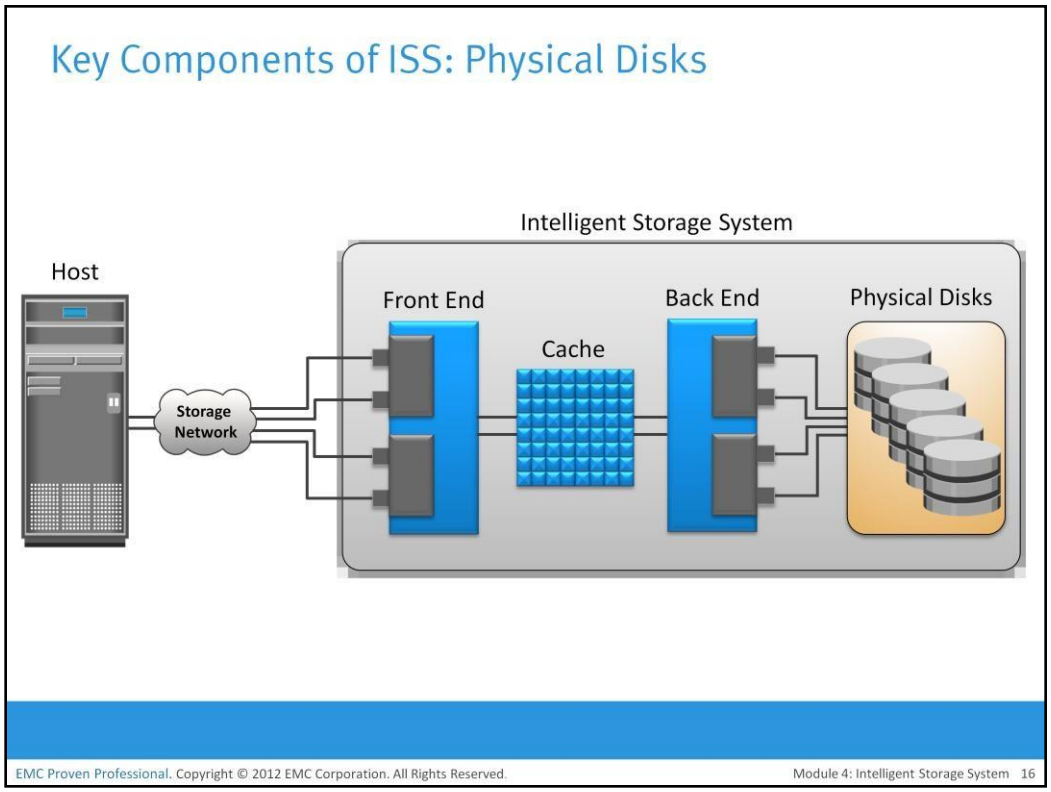
It intelligently determines which data would benefit by sitting in the server on PCIe flash and closer to the application. This avoids the latencies associated with I/O access over the network to the storage array. With this, the processing power required for an application's most frequently referenced data is offloaded from the back-end storage to the PCIe card.

Therefore, the storage array can allocate greater processing power to other applications.

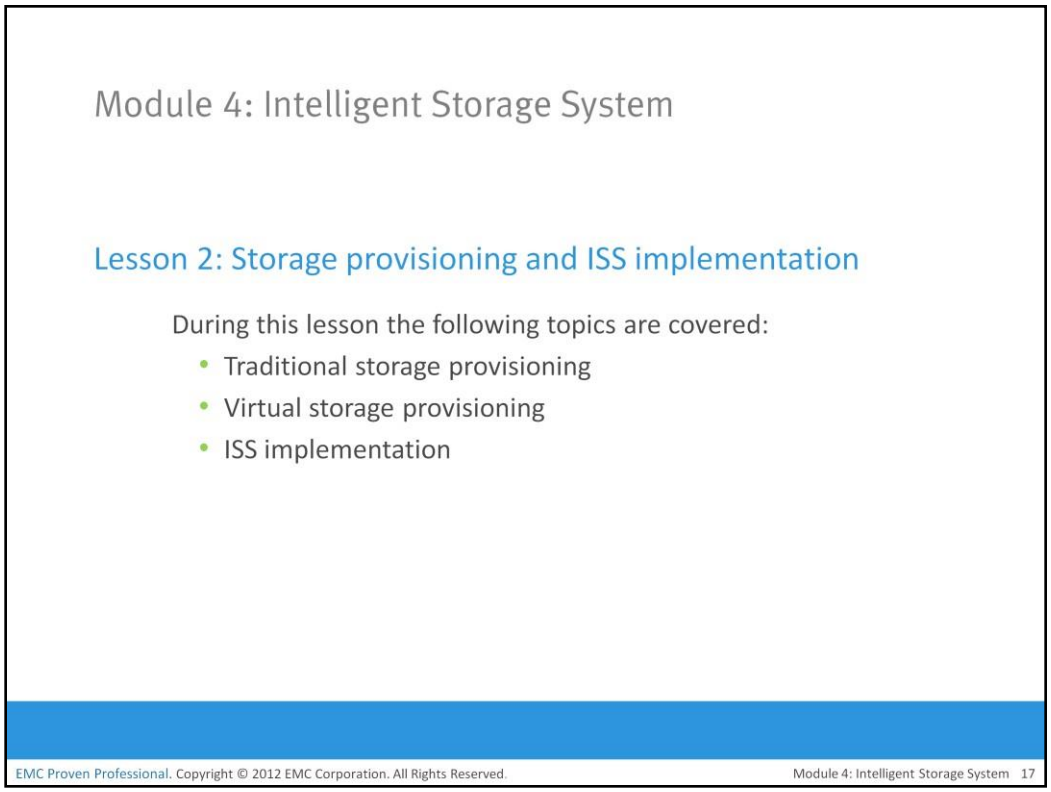


The *back end* provides an interface between cache and the physical disks. It consists of two components: back-end ports and back-end controllers. The back-end controls data transfers between cache and the physical disks. From cache, data is sent to the back end and then routed to the destination disk.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

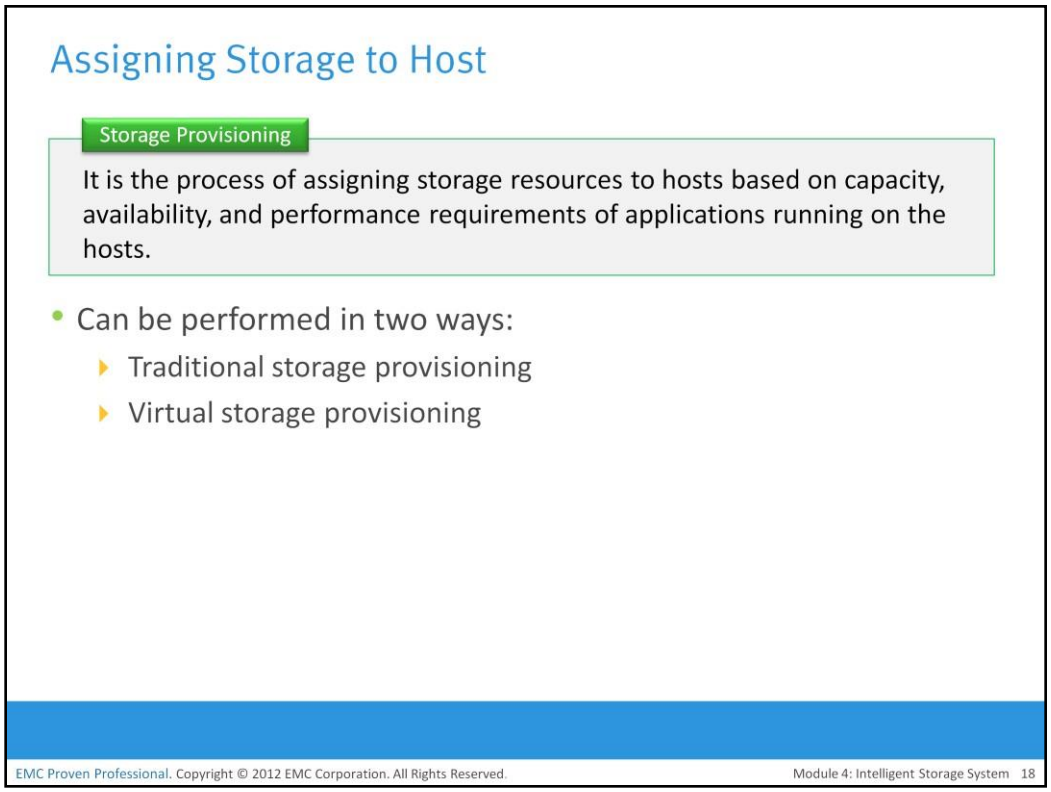
For high data protection and high availability, storage systems are configured with dual controllers with multiple ports. Such configurations provide an alternative path to physical disks if a controller or port failure occurs. This reliability is further enhanced if the disks are also dual-ported. In that case, each disk port can connect to a separate controller. Multiple controllers also facilitate load balancing.



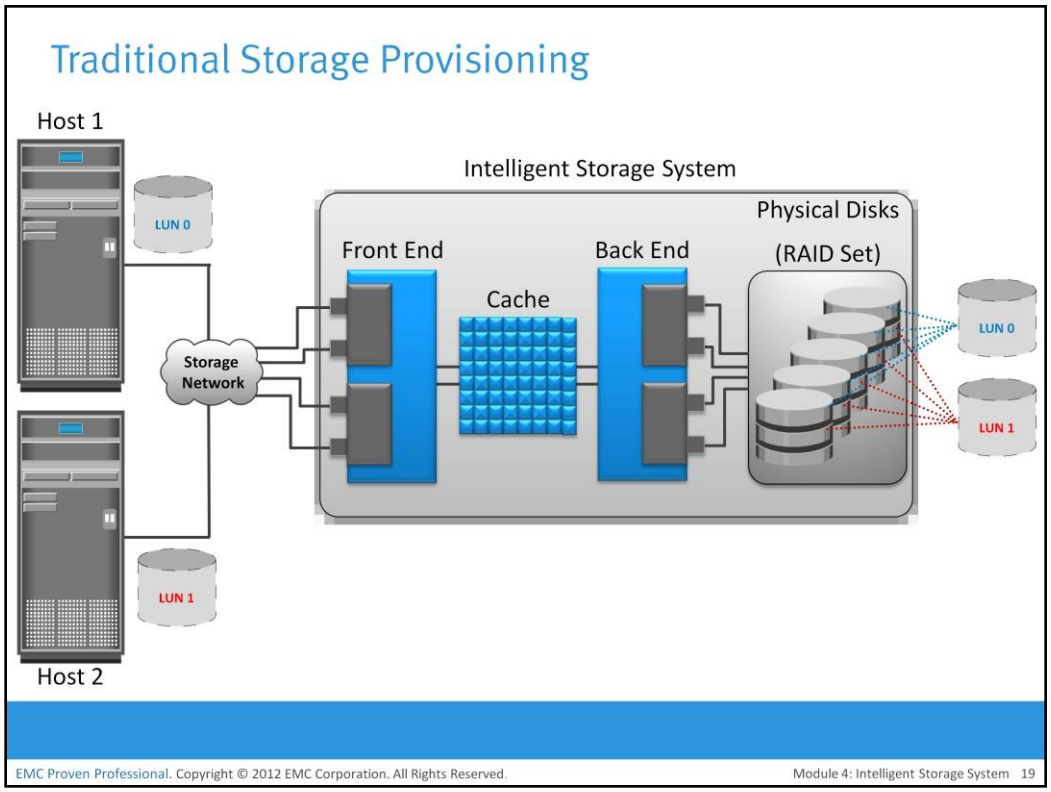
Physical disks are connected to the back-end storage controller and provide persistent data storage. Modern intelligent storage systems provide support to a variety of disk drives with different speeds and types, such as FC, SATA, SAS, and flash drives. They also support the use of a mix of flash, FC,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



This lesson focuses on traditional and virtual storage provisioning. This lesson also focuses on
ISS implementation.



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.



In traditional storage provisioning, physical disks are logically grouped together on which a required RAID level is applied to form a set, called 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.

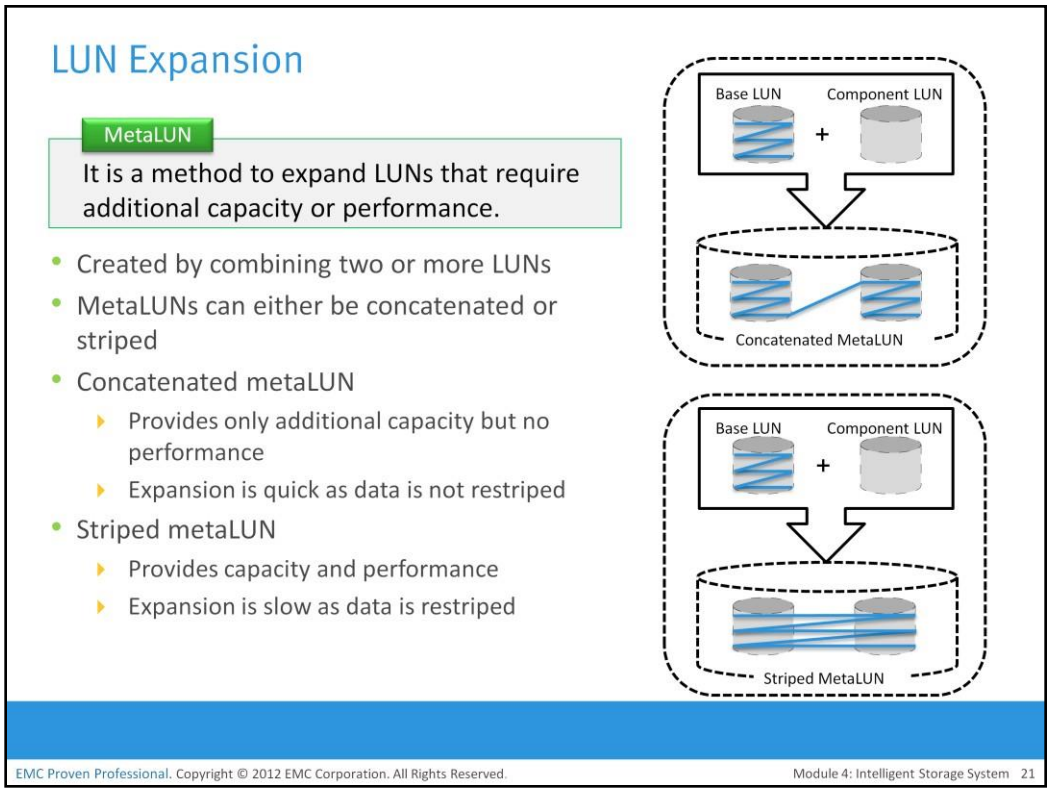
Figure on the slide shows a RAID set consisting of five disks that have been sliced, or partitioned, into two LUNs: LUN 0 and LUN 1. These LUNs are then assigned to Host1 and Host 2 for their storage requirements.

Cont..

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 file 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 files 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.

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.

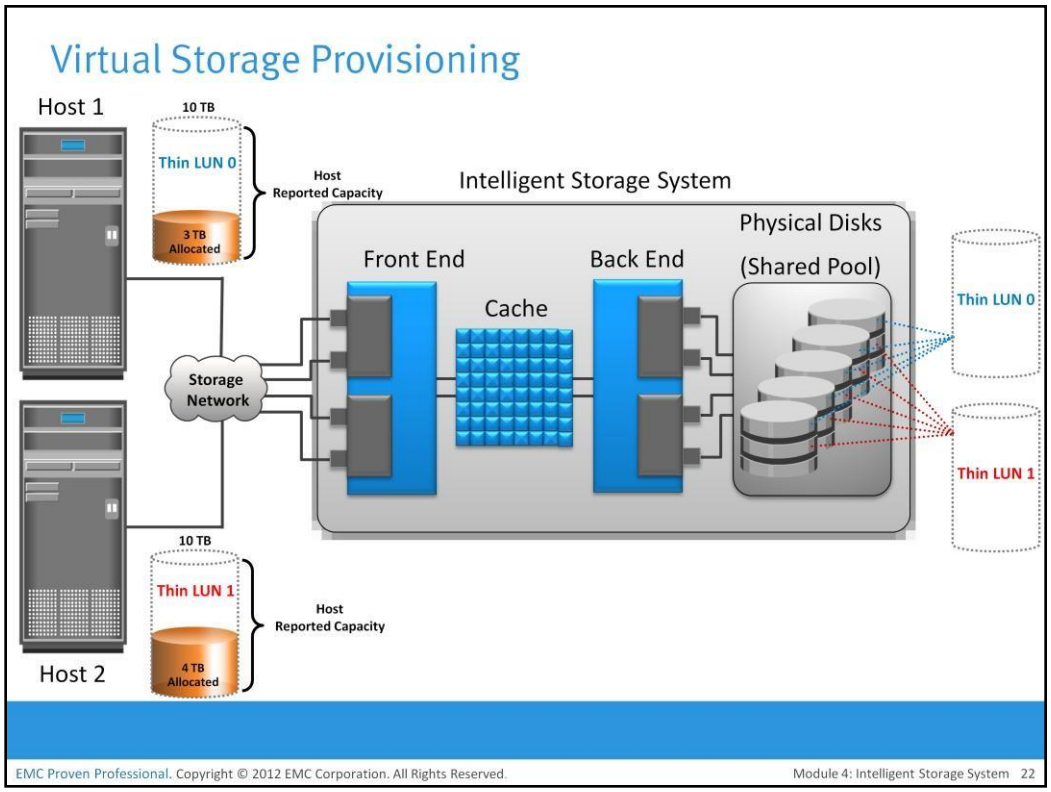


MetaLUN is a method to expand LUNs that require additional capacity or performance. A metaLUN can be created by combining two or more LUNs. A metaLUN consists of a base LUN and one or more component LUNs. MetaLUNs can be either *concatenated* or *striped*.

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.

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.

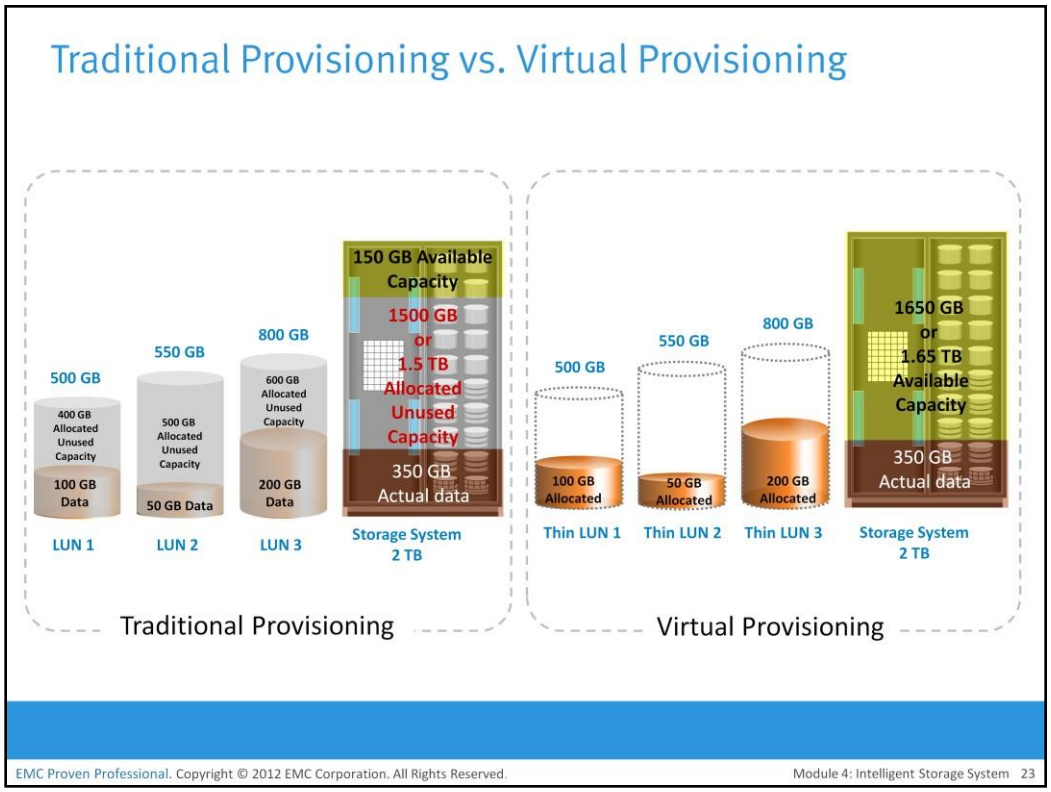
All LUNs in both concatenated and striped expansion must reside on the same disk-drive type: either all Fibre Channel or all ATA.



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* is 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.



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 on the slide illustrates an example, comparing virtual provisioning with traditional storage provisioning.

With traditional provisioning, three LUNs are created and presented to one or more hosts. 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.

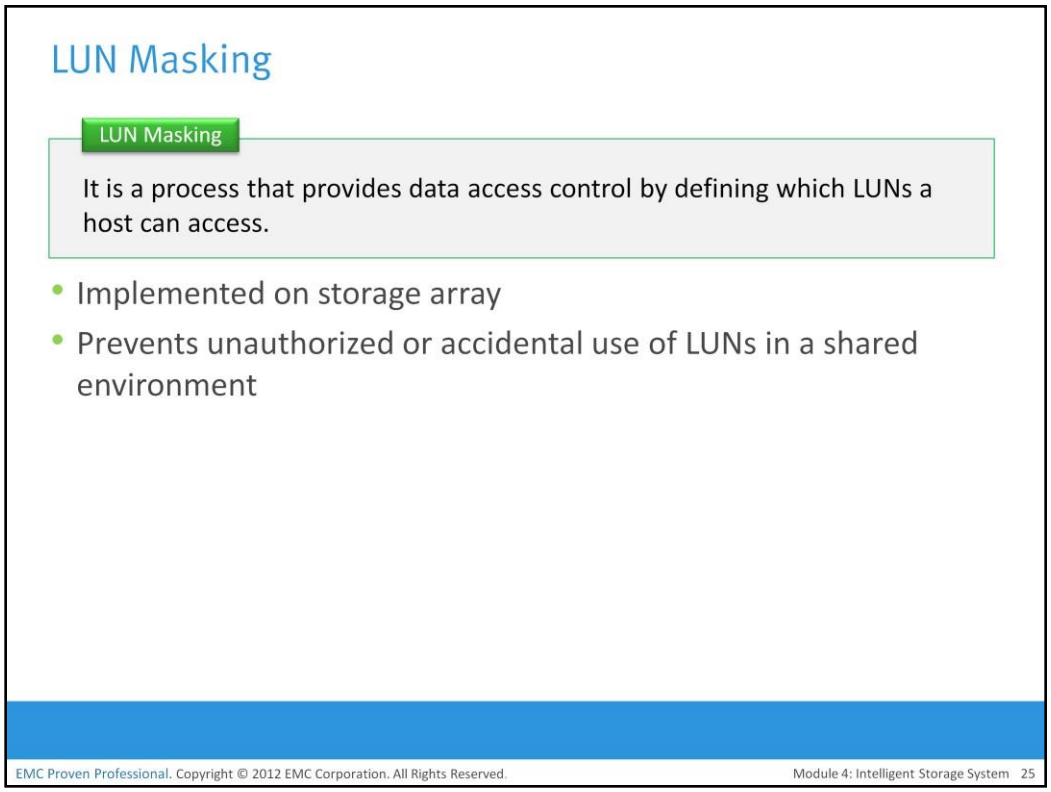
Now consider the same 2 TB storage system with virtual provisioning. Here, three thin LUNs of the same sizes are created. However, there is no allocated unused capacity. In total, the storage system with virtual provisioning has the same 350 GB of data, but 1.65 TB of capacity is available for other applications, whereas only 150 GB is available in traditional storage provisioning.

Cont..

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.

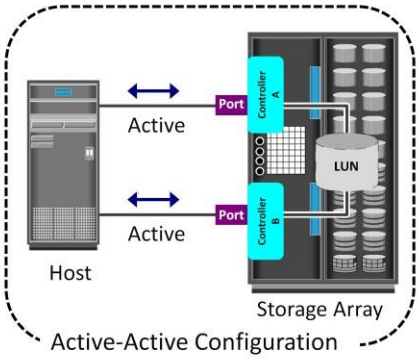


LUN masking is a process that provides data access control by defining which LUNs a host can access. The LUN masking function is implemented on the storage array. This ensures that volume access by hosts is controlled appropriately, preventing unauthorized or accidental use in a shared environment.

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 ISS: High-end Storage Systems

- Referred as active-active arrays, and generally aimed at large enterprise applications
  - Performs I/Os to LUNs through all the available paths
- These arrays provide the following capabilities:
  - Large storage capacity and cache
  - Fault tolerant architecture
  - Connectivity to mainframe and open systems
  - Multiple front-end ports and interface protocols
  - Ability to handle large amount of concurrent I/Os
  - Support local and remote data replication



Intelligent storage systems generally fall into one of the following two categories: high-end
storage systems, and midrange 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 .

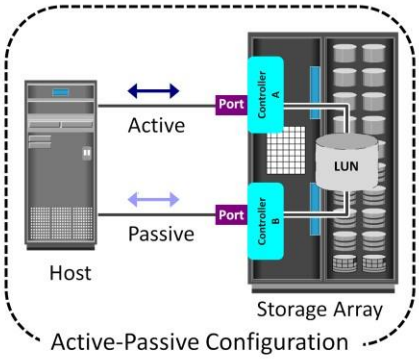
To address enterprise storage needs, these arrays provide the following capabilities:

- 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.

Types of ISS: Midrange Storage Systems

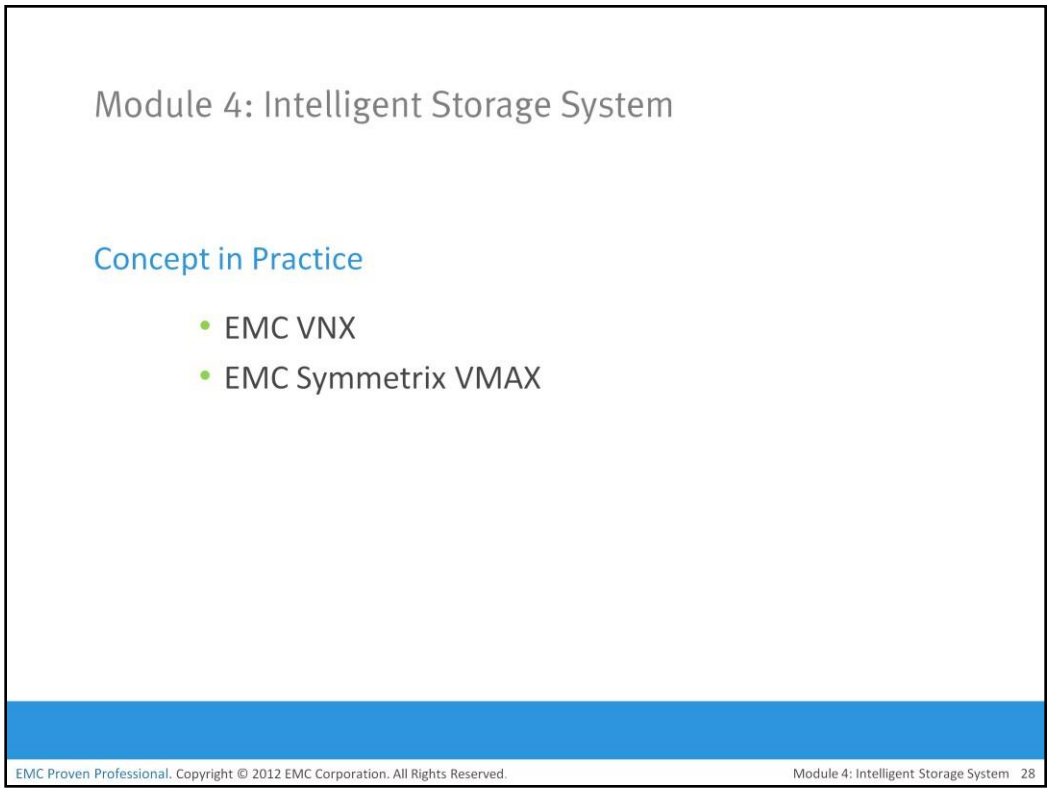
- Referred as active-passive arrays, and generally aimed at small and medium-sized enterprise applications
  - Performs I/Os to LUNs only through active paths
- These arrays typically have two controllers, each with cache, RAID controllers, and disks drive interfaces
- Less front-end ports, storage capacity, and cache as compared to high-end arrays
- Support local and remote data replication



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. 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.

Midrange arrays are designed to meet the requirements of small and medium enterprise application; 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.



The Concept in Practice covers the product example of intelligent storage system. It covers
two products: EMC Symmetrix and EMC VNX.

EMC VNX

- EMC's midrange storage offering
- Unified storage offering that provides storage for block, file, and object data
- Ideally suited for applications with predictable workloads

EMC VNX

The EMC VNX storage array is EMC's midrange storage offering that delivers enterprise- quality features and functionalities. EMC VNX is a unified storage platform that offers storage for block, file, and object-based data within the same array. It is ideally suited for applications with predictable workloads that require moderate-to-high throughput.

EMC Symmetrix VMAX

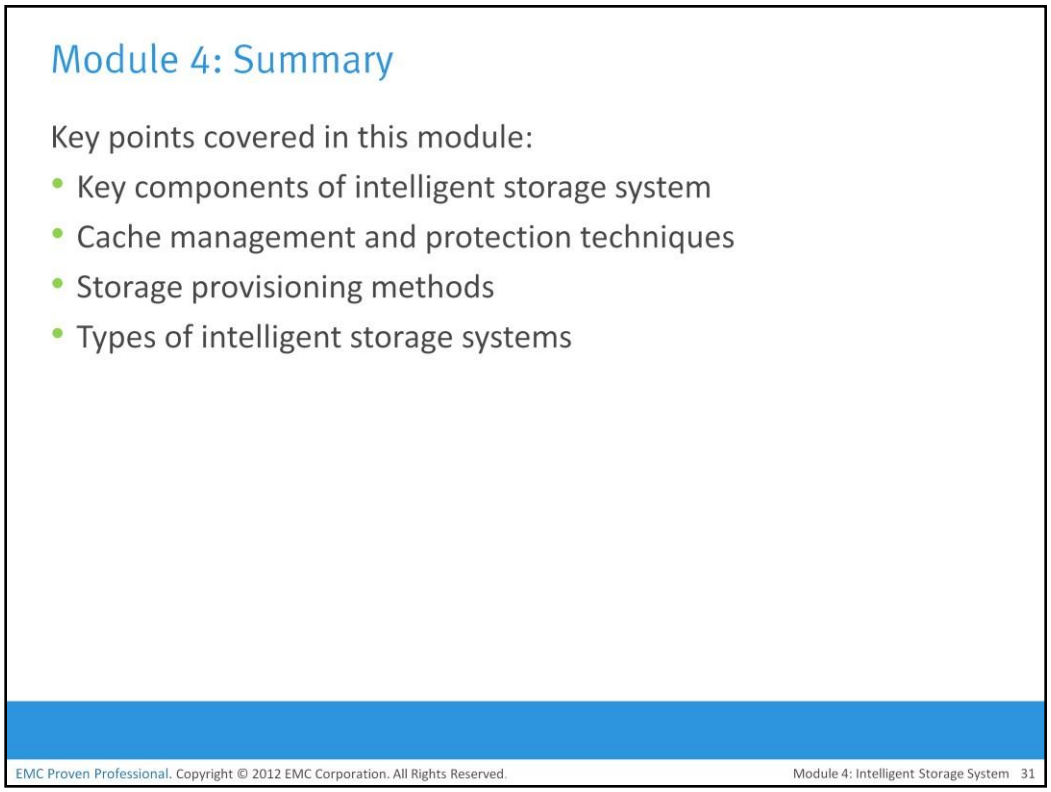
- EMC's high-end storage offering
- Key features supported by Symmetrix VMAX are:
  - Incrementally scalable to 2,400 disks
  - Supports up to 8 VMAX engines
  - Supports flash drives, fully automated storage tiering (FAST), virtual provisioning, and cloud computing
  - Supports up to 1 TB of global cache memory
  - Supports FC, iSCSI, GigE, and FICON for host connectivity
  - Supports RAID levels 1, 1+0, 5, and 6
  - Supports storage-based replication via EMC TimeFinder and SRDF

EMC Symmetrix VMAX

EMC Symmetrix establishes the highest standards for performance and capacity for an enterprise information storage solution and is recognized as the industry's most trusted storage platform. Symmetrix offers the highest level of scalability and performance to meet even unpredictable I/O workload requirements. The EMC Symmetrix offering includes Symmetrix Virtual Matrix (VMAX) series.

The EMC Symmetrix VMAX series is an innovative platform built around a scalable Virtual Matrix architecture to support the future storage growth demands of virtual IT environments. The key features supported by Symmetrix VMAX follows:

- Incrementally scalable to 2,400 disks
- Supports up to 8 VMAX engines (Each VMAX engine contains a pair of directors)
- Supports flash drives, fully automated storage tiering (FAST), virtual provisioning, and Cloud computing
- Supports up to 1 TB of global cache memory
- Supports FC, iSCSI, GigE, and FICON for host connectivity
- Supports RAID levels 1, 1+0, 5, and 6
- Supports storage-based replication through EMC TimeFinder and EMC SRDF
- Highly fault-tolerant design that allows nondisruptive upgrades and full component-level redundancy with hot-swappable replacements



This module covered the four key components of intelligent storage systems, front end, cache, back end, and physical disks. Further, this module detailed the cache management and protection techniques such as flushing, least recently used, and most recently used algorithm. It also covered the two storage provisioning techniques, traditional and virtual storage provisioning. Finally, the module described the two types of intelligent storage systems such as high-end and midrange storage systems.

Check Your Knowledge – 1

- Which component of an intelligent storage system isolates host from the mechanical delays associated with rotating disks?
  - A. Front-end controller
  - B. Back-end controller
  - C. Cache
  - D. Storage network
- Which mode of flushing is activated when the cache reaches 100% of its capacity?
  - A. Idle
  - B. High watermark
  - C. Forced
  - D. Low watermark

Check Your Knowledge – 2

- In traditional storage provisioning, which LUN expansion technique provides improved performance?
  - A. Concatenated metaLUN
  - B. Striped metaLUN
  - C. Base LUN
  - D. Component LUN
- Which process provides data access control by restricting host access to specific LUN(s)?
  - A. LUN masking
  - B. Zoning
  - C. Trespassing
  - D. VSAN

Check Your Knowledge – 3

- Which mechanism provides protection to 'uncommitted data in cache' against power failure?
  - A. Mirroring
  - B. Vaulting
  - C. Watermarking
  - D. Tiering