Welcome to Intelligent Storage Systems. This module is part of ISM v3 course that includes 16 modules.

Revision Date: August 2015

Revision Number: MR-1WP-ISMv3ISS
This module covers the key components of an intelligent storage system. It also covers storage subsystems, and the components, addressing, and performance parameters of hard disk drive and solid state drive. This module also covers RAID, various RAID implementations, techniques, and commonly-used RAID levels. Further, this module covers the impact of RAID on performance, and compares the commonly-used RAID levels. Finally, this module covers the types of intelligent storage systems and their architectures.
Third Platform Requirements for Storage

- Process massive amount of IOPS
- Elastic and non-disruptive horizontal scaling of resources
- Intelligent resource management
- Automated and policy driven configuration
- Support for multiple protocols for data access
- Supports APIs for software-defined and cloud integration
- Centralized management and chargeback in a multi-tenancy environment

The slide provides a list of key requirements for an effective storage infrastructure.
Technology Solution

- Intelligent storage system
  - Block-based storage system
  - File-based storage system
  - Object-based storage system
  - Unified storage system
- Storage Virtualization
- Software-defined storage

The slide lists technology solutions that can meet the third platform requirements for the storage infrastructure.
This lesson covers components of intelligent storage systems. This lesson also covers components, addressing, and performance of hard disk drives.
Intelligent storage systems are feature-rich RAID arrays that provide highly optimized I/O processing capabilities. These intelligent storage systems have the capability to meet the requirements of today’s I/O intensive third platform applications. These applications require high levels of performance, availability, security, and scalability. Therefore, to meet the requirements of the applications many vendors of intelligent storage systems now support SSDs, encryption, compression, deduplication, and scale-out architecture. The use of SSDs and scale-out architecture enable to service massive number of IOPS. These storage systems also support connectivity to heterogeneous compute systems.

The storage systems have an operating environment that intelligently and optimally handles the management, provisioning, and utilization of storage resources. The 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.

The storage systems also support various technologies, such as deduplication, compression, encryption, automated storage tiering, and virtual storage provisioning. These capabilities have added a new dimension to storage system performance, scalability, and availability. Further, the intelligent storage systems support APIs to enable integration with SDDC and cloud environments.
An intelligent storage system has two key components, controller and storage. A controller is a compute system that runs a purpose-built operating system that is responsible for performing several key functions for the storage system. Examples of such functions are serving I/Os from the application servers, storage management, RAID protection, local and remote replication, provisioning storage, automated tiering, data compression, data encryption, and intelligent cache management.

An intelligent storage system typically has more than one controller for redundancy. Each controller consists of one or more processors and a certain amount of cache memory to process a large number of I/O requests. These controllers are connected to the compute system either directly or via a storage network. The controllers receive I/O requests from the compute systems that are read or written from/to the storage by the controller. Depending on the type of the data access method (discussed later in this module) used for a storage system, the controller can either be classified as block-based, file-based, object-based, or unified.

An intelligent storage system can have all hard disk drives, all solid state drives, or a combination of both.
A hard disk drive is a persistent storage device that stores and retrieves data using rapidly rotating disks (platters) coated with magnetic material. The key components of a hard disk drive (HDD) are platter, spindle, read-write head, actuator arm assembly, and controller board. I/O operations in an HDD are performed by rapidly moving the arm across the rotating flat platters coated with magnetic material. Data is transferred between the disk controller and magnetic platters through the read-write (R/W) head which is attached to the arm. Data can be recorded and erased on magnetic platters any number of times.

- **Platter:** A typical HDD consists of one or more flat circular disks called platters. The data is recorded on these platters in binary codes (0s and 1s). The set of rotating platters is sealed in a case, called Head Disk Assembly (HDA). A platter is a rigid, round disk coated with magnetic material on both surfaces (top and bottom). The data is encoded by polarizing the magnetic area or domains of the disk surface. Data can be written to or read from both surfaces of the platter. The number of platters and the storage capacity of each platter determine the total capacity of the drive.

- **Spindle:** A spindle connects all the platters and is connected to a motor. The motor of the spindle rotates with a constant speed. The disk platter spins at a speed of several thousands of revolutions per minute (rpm). Common spindle speeds are 5,400 rpm, 7,200 rpm, 10,000 rpm, and 15,000 rpm. The speed of the platter increases with the improvement in technology; although the extent to which it can be improved is limited.

(Cont’d)
• **Read/write head:** Read/write (R/W) heads, read and write data from or to the platters. Drives have two R/W heads per platter, one for each surface of the platter. The R/W head changes the magnetic polarization on the surface of the platter when writing data. While reading data, the head detects the magnetic polarization on the surface of the platter. During reads and writes, the R/W head senses the magnetic polarization and never touches the surface of the platter. When the spindle rotates, a microscopic air gap is maintained between the R/W heads and the platters, known as the head flying height. This air gap is removed when the spindle stops rotating and the R/W head rests on a special area on the platter near the spindle. This area is called the landing zone. The landing zone is coated with a lubricant to reduce friction between the head and the platter. The logic on the disk drive ensures that heads are moved to the landing zone before they touch the surface. If the drive malfunctions and the R/W head accidentally touches the surface of the platter outside the landing zone, a head crash occurs. In a head crash, the magnetic coating on the platter is scratched and may cause damage to the R/W head. A head crash generally results in data loss.

• **Actuator arm assembly:** R/W heads are mounted on the actuator arm assembly, which positions the R/W head at the location on the platter where the data needs to be written or read. The R/W heads for all platters on a drive are attached to one actuator arm assembly and move across the platters simultaneously.

• **Drive controller board:** The controller is a printed circuit board, mounted at the bottom of a disk drive. It consists of a microprocessor, internal memory, circuitry, and firmware. The firmware controls the power supplied to the spindle motor as well as controls the speed of the motor. It also manages the communication between the drive and the compute system. In addition, it controls the R/W operations by moving the actuator arm and switching between different R/W heads, and performs the optimization of data access.
Data on the disk is recorded on tracks, which are concentric rings on the platter around the spindle. The tracks are numbered, starting from zero, from the outer edge of the platter. The number of tracks per inch (TPI) on the platter (or the track density) measures how tightly the tracks are packed on a platter.

Each track is divided into smaller units called sectors. A sector is the smallest, individually addressable unit of storage. The track and sector structure is written on the platter by the drive manufacturer using a low-level formatting operation. The number of sectors per track varies according to the drive type. There can be thousands of tracks on a platter, depending on the physical dimensions and the recording density of the platter.

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

A cylinder is a set of identical tracks on both surfaces of each drive platter. The location of R/W heads is referred to by the cylinder number, not by the track number.
The earlier drives used physical addresses consisting of cylinder, head, and sector (CHS) number to refer to specific locations on the disk, and the OS had to be aware of the geometry of each disk used. **Logical block addressing** (LBA) has simplified the addressing by using a linear address to access physical blocks of data. The disk controller translates LBA to a CHS address, and the compute system needs to know only the size of the disk drive in terms of the number of blocks. The logical blocks are mapped to physical sectors on a 1:1 basis.

In the slide, the drive shows eight sectors per track, six heads, and four cylinders. This means a total of $8 \times 6 \times 4 = 192$ blocks; so the block number ranges from 0 to 191. Each block has its own unique address.

Assuming that the sector holds 512 bytes, a 500 GB drive with a formatted capacity of 465.7 GB has in excess of 976,000,000 blocks.
A disk drive is an electromechanical device that governs the overall performance of the storage system environment. The various factors that affect the performance of disk drives are:

- Seek time
- Rotational latency
- Data transfer rate
Seek Time

- Time taken to position the read/write head
- The lower the seek time, the faster the I/O operation
- Seek time specifications include
  - Full stroke
  - Average
  - Track-to-track
- The seek time of a disk is specified by the drive manufacturer

The seek time (also called access time) describes the time taken to position the R/W heads across the platter with a radial movement (moving along the radius of the platter). In other words, it is the time taken to position and settle the arm and the head over the correct track. Therefore, the lower the seek time, the faster the I/O operation. Disk vendors publish the following seek time specifications:

- **Full Stroke**: It is the time taken by the R/W head to move across the entire width of the disk, from the innermost track to the outermost track.

- **Average**: It is the average time taken by the R/W head to move from one random track to another, normally listed as the time for one-third of a full stroke.

- **Track-to-Track**: It is the time taken by the R/W head to move between adjacent tracks.

Each of these specifications is measured in milliseconds (ms). The seek time of a disk is typically specified by the drive manufacturer. The average seek time on a modern disk is typically in the range of 3 to 15 ms. Seek time has more impact on the I/O operation of random tracks rather than the adjacent tracks. To minimize the seek time, data can be written to only a subset of the available cylinders. This results in lower usable capacity than the actual capacity of the drive. For example, a 500 GB disk drive is set up to use only the first 40 percent of the cylinders and is effectively treated as a 200 GB drive. This is known as short-stroking the drive.
Rotational Latency

- The time taken by the platter to rotate and position the data under the R/W head
- Depends on the rotation speed of the spindle
- Average rotational latency
  - One-half of the time taken for a full rotation
  - For ‘X’ rpm, drive latency is calculated in milliseconds as:

\[
\frac{1}{2} \times 1000 \left(\frac{X}{60}\right) = \frac{500}{X} = \frac{30000}{X}
\]

To access data, the actuator arm moves the R/W head over the platter to a particular track while the platter spins to position the requested sector under the R/W head. The time taken by the platter to rotate and position the data under the R/W head is called rotational latency. This latency depends on the rotation speed of the spindle and is measured in milliseconds. The average rotational latency is one-half of the time taken for a full rotation. Similar to the seek time, rotational latency has more impact on the reading/writing of random sectors on the disk than on the same operations on adjacent sectors.

Average rotational latency is approximately 5.5 ms for a 5,400-rpm drive, and around 2 ms for a 15,000-rpm drive as shown below.

Average rotational latency for 15K rpm (30000/15000) drive is \( \frac{30000}{15000} = 2 \text{ ms} \)
The **data transfer rate** (also called **transfer rate**) refers to the average amount of data per unit time that the drive can deliver to the HBA. In a **read operation**, the data first moves from disk platters to R/W heads; then it moves to the drive’s internal **buffer**. Finally, data moves from the buffer through the interface to the compute system’s HBA. In a **write operation**, the data moves from the HBA to the internal buffer of the disk drive through the drive’s interface. The data then moves from the buffer to the R/W heads. Finally, it moves from the R/W heads to the platters. The data transfer rates during the R/W operations are measured in terms of internal and external transfer rates, as shown on the slide.

**Internal transfer rate** is the speed at which data moves from a platter’s surface to the internal buffer (cache) of the disk. The internal transfer rate takes into account factors such as the seek time and rotational latency. **External transfer rate** is the rate at which data can move through the interface to the HBA. The external transfer rate is generally the advertised speed of the interface, such as 133 MB/s for ATA.
The utilization of a disk I/O controller has a significant impact on the I/O response time. Consider that a disk is viewed as a black box consisting of two elements: the queue and the disk I/O controller. Queue is the location where an I/O request waits before it is processed by the I/O controller and disk I/O controller processes I/Os waiting in the queue one by one.

The I/O requests arrive at the controller at the rate generated by the application. The I/O arrival rate, the queue length, and the time taken by the I/O controller to process each request determines the I/O response time. If the controller is busy or heavily utilized, the queue size will be large and the response time will be high. Based on the fundamental laws of disk drive performance, the relationship between controller utilization and average response time is given as:

$$\text{Average response time} = \frac{\text{Service time}}{(1 - \text{Utilization})}$$

where, service time is the time taken by the controller to serve an I/O.

As the utilization reaches 100 percent, that is, as the I/O controller saturates, the response time moves closer to infinity. In essence, the saturated component or the bottleneck forces the serialization of I/O requests; meaning, each I/O request must wait for the completion of the I/O requests that preceded it. The figure on the slide shows a graph plotted between utilization and response time. The graph indicates that as the utilization increases, the response time changes are nonlinear. When the average queue sizes are low, the response time remains low. The response time increases slowly with added load on the queue and increases exponentially when the utilization exceeds 70 percent. Therefore, for performance-sensitive applications, it is common to utilize disks below their 70 percent of I/O serving capability.
Determining storage requirements for an application begins with determining the required storage capacity and I/O performance. Capacity can be easily estimated by the size and number of file systems and database components used by applications. The I/O size, I/O characteristics, and the number of I/Os generated by the application at peak workload are other factors that affect performance, I/O response time and design of storage system.

The disk service time ($T_S$) for an I/O is a key measure of disk performance; $T_S$, along with disk utilization rate ($U$), determines the I/O response time for an application. As discussed earlier the total disk service time is the sum of the seek time, rotational latency, and transfer time.

Note that transfer time is calculated based on the block size of the I/O and given data transfer rate of a disk drive. For example, for an I/O with a block size of 32 KB and given disk data transfer rate of 40MB/s; the transfer time will be 32 KB/40 MB.

$T_S$ determines the time taken by the I/O controller to serve an I/O, therefore, the maximum number of I/Os serviced per second or IOPS is $(1/T_S)$.

The IOPS calculated above represents the IOPS that can be achieved at potentially high levels of I/O controller utilization (close to 100 percent). If the application demands a faster response time, then the utilization for the disks should be maintained below 70 percent.

Based on this discussion, the total number of disks required for an application is computed as: Max (Disks required for meeting capacity, Disks required for meeting performance)
Consider an example in which the capacity requirement for an application is 1.46 TB. The number of IOPS generated by the application at peak workload is estimated at 9,000 IOPS. The vendor specifies that a 146 GB, 15,000-rpm drive is capable of doing a maximum of 180 IOPS.

In this example, the number of disks required to meet the capacity requirements will be 1.46 TB/146 GB = 10 disks.

To meet the application IOPS requirements, the number of disks required is 9,000 / 180 = 50. However, if the application is response-time sensitive, the number of IOPS a disk drive can perform should be calculated based on 70 percent disk utilization. Considering this, the number of IOPS a disk can perform at 70 percent utilization is 180 x 0.7 = 126 IOPS. Therefore, the number of disks required to meet the application IOPS requirement will be 9,000/126 = 72.

As a result, the number of disks required to meet the application requirements will be Max (10, 72) = 72 disks.

The preceding example indicates that from a capacity perspective, 10 disks are sufficient; however, the number of disks required to meet application performance is 72. To optimize disk requirements from a performance perspective, various solutions are deployed in a real-time environment. Examples of these solutions are disk native command queuing, use of flash drives, RAID, and the use of cache memory. RAID and cache are detailed in module 5, ‘Block-based Storage System’.
Lesson 1: Summary

During this lesson the following topics were covered:

- Components of intelligent storage systems
- HDD components, addressing, and performance

This lesson covered the components of intelligent storage systems. This lesson also covered the components, addressing, and performance of HDDs.
Lesson 2: Components of Intelligent Storage Systems – II

This lesson covers the following topics:

• SSD components, addressing, and performance

This lesson covers components, addressing, and performance of solid state drives.
Solid state drives (SSDs) are storage devices that contain non-volatile flash memory. Solid state drives are superior to mechanical hard disk drives in terms of performance, power use, and availability. These drives are especially well suited for low-latency applications that require consistent, low (less than 1 ms) read/write response times. In a HDD servicing, small-block, highly-concurrent, random workloads involve considerable rotational and seek latency, which significantly reduces throughput.

Externally solid state drives have the same physical format and connectors as mechanical hard disk drives. This maintains the compatibility in form and format with mechanical hard disk drives, and allows easy replacement of a mechanical drive with a solid state drive. Internally, a solid state drive’s hardware architecture consists of the following components: I/O interface, controller, and mass storage.

The I/O interface enables connecting the power and data connectors to the solid state drives. SSDs typically support standard connectors such as SATA, SAS, or FC.

The controller includes a drive controller, RAM, and non-volatile memory (NVRAM). The drive controller manages all drive functions. The SSDs include many features such as encryption and write coalescing. The non-volatile RAM (NVRAM) is used to store the SSD’s operational software and data. Not all SSDs have separate NVRAM. Some models store their programs and data to the drive’s mass storage. The RAM is used in the management of data being read and written from the SSD as a cache, and for the SSD’s operational programs and data. The portion of the drive’s RAM used for controller cache enhances the overall performance of the SSD. Mass storage, which is made of flash memories, writes slower than it reads. The drive’s RAM is used to minimize the number of writes to mass storage and improve the response time of the drive.

(Cont’d)
Write coalescing is one of the techniques employed within the RAM. This is the process of grouping write I/Os and writing them in a single internal operation versus many smaller-sized write operations. In addition to caching, the RAM contains the drive controller’s operational software and mapping tables. Mapping tables correlate the internal data structure of the SSD to the file system data structure of the compute system.

The mass storage is an array of non-volatile memory chips. They retain their contents when powered off. These chips are commonly called Flash memory. The number and capacity of the individual chips vary directly in relationship to the SSD’s capacity. The larger the capacity of the SSD, the larger is the capacity and the greater is the number of the Flash memory chips.

The Flash memory chips that make up the drive’s mass storage come from numerous manufacturers. Two types of Flash memory chip are used in commercially available SSDs: Single-Level Cell (SLC) and Multi-Level Cell (MLC). SLC-type Flash is typically used in enterprise-rated SSDs for its increased memory speed and longevity. MLC is slower but has the advantage of greater capacity per chip. Although SLC type Flash memory offers a lower density, it also provides a higher level of performance in the form of faster reads and writes. In addition, SLC Flash memory has higher reliability. As SLC Flash memory stores only one bit per cell, the likelihood for error is reduced. SLC also allows for higher write/erase cycle endurance. For these reasons, SLC Flash memory is preferred for use in applications requiring higher reliability, and increased endurance and viability in multi-year product life cycles.

SSDs consume less power compared to hard disk drives. Because SSDs do not have moving parts, they generate less heat compared to HDDs. Therefore, it further reduces the need for cooling in storage enclosure, which further reduces the overall system power consumption.

SSDs have multiple parallel I/O channels from its drive controller to the flash memory storage chips. Generally, the larger the number of flash memory chips in the drive, the larger is the number of channels. The larger the number of channels, the greater is the SSD’s internal bandwidth. The drive’s controller uses native command queuing to efficiently distribute read and write operations across all available channels. Bandwidth performance scales upward with parallel use of all available channels. Note that the drives with the same capacity, but from different vendors, can have a different number of channels. These drives will have different levels of performance. The drive with more channels will outperform the drive with a fewer under some circumstances.
Solid state memory chips have different capacities, for example a solid state memory chip can be 32 GB or 4 GB per chip. However, all memory chips share the same logical organization, that is pages and blocks.

At the lowest level, a solid state drive stores bits. Eight bits make up a byte, and while on the typical mechanical hard drive 512 bytes would make up a sector, solid state drives do not have sectors. Solid state drives have a similar physical data object called a page. Like a mechanical hard drive sector, the page is the smallest object that can be read or written on a solid state drive. Unlike mechanical hard drives, pages do not have a standard capacity. A page’s capacity depends on the architecture of the solid state memory chip. Typical page capacities are 4 KB, 8 KB, and 16 KB.

A solid state drive block is made up of pages. A block may have 32, 64, or 128 pages. 32 is a common block size. The total capacity of a block is dependent on the solid state chip’s page size. Only entire blocks may be written or erased on a solid state memory chip. Individual pages may be read or invalidated (a logical function). For a block to be written, pages are assembled into full blocks in the solid state drive’s cache RAM and then written to the block storage object.
A page has three possible states, erased (empty), valid, and invalid. In order to write any data to a page, its owning block location on the flash memory chip must be electrically erased. This function is performed by the SSD’s hardware. Once a page has been erased, new data can be written to it. For example, when a 4 KB of data is written to a 4 KB capacity page, the state of that page is changed to valid, as it is holding valid data. A valid page’s data can be read any number of times. If the drive receives a write request to the valid page, the page is marked invalid and that write goes to another page. This is because erasing blocks is time consuming and may increase the response time. Once a page is marked invalid, its data can no longer be read. An invalid page needs to be erased before it can once again be written with new data. Garbage collection handles this process. Garage collection is the process of providing new erased blocks.

A block has three possible states, erased (empty), new, and used. Once a block is erased, a block’s number of pages that have been assembled in the SSD’s RAM may be written to it. For example, thirty two 4 KB pages may be assembled into a block, and then written to the erased block. This sets the block’s state to “new”, meaning it is holding pages with valid data. A block’s valid pages can be read any number of times. There are two mechanisms to invalidate a page, writes and deletes. If the drive receives a write request to a valid block page, the page must be changed. The current page containing the destination of the write is marked invalid. The block’s state changes to “used”, because it contains invalid pages. These writes go to another page, on an erased block. A delete invalidates a page without resulting in a subsequent write.
Solid state drives are semiconductor, random-access devices; these result in very low response times compared to hard disk drives. This, combined with the multiple parallel I/O channels on the back end, gives SSDs performance characteristics that are better than HDDs.

SSD performance is dependent on access type, drive state, and workload duration. SSD performs random reads the best. In carefully tuned multi-threaded, small-block random I/O workload storage environments, SSDs can deliver much lower response times and higher throughput than HDDs. This is because random-read I/Os cannot usually be serviced by read-ahead algorithms on a HDD or by read cache on the storage system. The latency of a random read operation is directly related to the seek time of a HDD. For HDDs, this is the physical movement of the drive’s read/write head to access the desired area. Because they are random access devices, SSDs pay no penalty for retrieving I/O that is stored in more than one area; as a result their response time is in an order of magnitude faster than the response time of HDDs.

For large block I/Os, SSDs tend to use all internal I/O channels in parallel. Since the single-threaded sequential I/O streams on FC HDDs do not suffer seek and rotational latencies (because of the storage system cache), single-threaded large-block sequential I/O streams will not show major performance improvements with SSDs over FC HDDs. However, with the increased application concurrency (as more threads are added), the load starts to resemble a large block-random workload. In this case, seek and rotational latencies are introduced that decrease the FC HDD effectiveness but do not decrease SSD effectiveness.

A new SSD or an SSD with substantial unused capacity has the best performance. Drives with substantial amounts of their capacity consumed will take longer to complete the read-modify-write cycle. SSDs are best for workloads with short bursts of activity.
Lesson 2: Summary

During this lesson the following topics were covered:

- SSD components
- SSD addressing
- SSD performance

This lesson covered the components, addressing, and performance of solid state drives.
Lesson 3: RAID

This lesson covers the following topics:

- Describe RAID implementation methods
- Describe the three RAID techniques
- Describe commonly used RAID levels
- Describe the impact of RAID on performance
- Compare RAID levels based on their cost, performance, and protection

This lesson covers RAID and its use to improve performance and protection. It covers various RAID implementations, techniques, and levels commonly used. This lesson also covers the impact of RAID on performance and compares the commonly used RAID levels.
Redundant Array of Independent Disks (RAID) is a technique in which multiple disk drives are combined into a logical unit called a RAID set and data is written in blocks across the disks in the RAID set. RAID protects against data loss when a drive fails, through the use of redundant drives and parity. RAID also helps in improving the storage system performance as read and write operations are served simultaneously from multiple disk drives.

RAID is typically implemented by using a specialized hardware controller present either on the compute system or on the storage system. The key functions of a RAID controller are management and control of drive aggregations, translation of I/O requests between logical and physical drives, and data regeneration in the event of drive failures.

(Cont’d)
There are two methods of RAID implementation, hardware and software. Both have their advantages and disadvantages.

*Software RAID* uses compute system-based software to provide RAID functions and is implemented at the operating-system level. Software RAID implementations offer cost and simplicity benefits when compared with hardware RAID. However, they have the following limitations:

- **Performance:** Software RAID affects the 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 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.

In hardware RAID implementations, a specialized hardware controller is implemented either on the compute system or on the storage system. Controller card RAID is a compute system-based hardware RAID implementation in which a specialized RAID controller is installed in the compute system, and disk drives are connected to it. Manufacturers also integrate RAID controllers on motherboards. A compute system-based RAID controller is not an efficient solution in a data center environment with a large number of compute systems. The external RAID controller is a storage system-based hardware RAID. It acts as an interface between the compute system and the disks. It presents storage volumes to the compute system, and the compute system 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
A RAID array is an enclosure that contains a number of disk drives and supporting hardware to implement RAID. A subset of disks within a RAID array can be grouped to form logical associations called logical arrays, also known as a RAID set or a RAID group.
The three different RAID techniques that form the basis for defining various RAID levels are striping, mirroring, and parity. These techniques determine the data availability and performance of a RAID set as well as the relative cost of deploying a RAID level.

Striping is a technique of spreading data across multiple drives (more than one) in order to use the drives in parallel. All the read-write heads work simultaneously, allowing more data to be processed in a shorter time and increasing performance, compared to reading and writing from a single disk. Within each disk in a RAID set, a predefined number of contiguously addressable disk blocks are defined as strip. The set of aligned strips that spans across all the disks within the RAID set is called a stripe. The figure 1 on the slide shows representations of a striped RAID set. Strip size (also called stripe depth) describes the number of blocks in a strip (represented as “A1, A2, A3, and A4”), 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 the data is broken into smaller pieces while it is spread across the disks. Stripe size (represented as A) is a multiple of strip size by the number of data disks in the RAID set. For example, in a four disk striped RAID set with a strip size of 64KB, the stripe size is 256 KB (64KB x 4). In other words, A = A1 +A2 + A3 + A4. 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.

(Cont’d)
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 remains intact on the surviving disk drive and the controller continues to service the compute system’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 compute system. 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.

**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. The first three disks in the figure, labeled $D_1 \text{ to } D_3$, contain the data. The fourth disk, labeled $P$, stores the parity information, which, in this case, is the sum of the elements in each row. Now, if one of the 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.

Compared to mirroring, parity implementation considerably reduces the cost associated with data protection. Consider an example of a parity RAID configuration with four disks where three disks hold data, and the fourth holds the parity information. In this example, parity requires only 33 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 four ($3 + 1$) disk parity RAID set with a strip size of 64 KB, the stripe size will be 192 KB (64KB x 3).
RAID Levels

- Commonly used RAID levels are:
  - RAID 0 – Striped set with no fault tolerance
  - RAID 1 – Disk mirroring
  - RAID 1 + 0 – Nested RAID
  - RAID 3 – Striped set with parallel access and dedicated parity disk
  - RAID 5 – Striped set with independent disk access and a distributed parity
  - RAID 6 – Striped set with independent disk access and dual distributed parity

The RAID level selection depends on the parameters such as application performance, data availability requirements, and cost. 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. The commonly used RAID levels are listed on the slide.
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 gathered by the controller. When the number of drives in the RAID set increases, the performance improves because more data can be read or written simultaneously. RAID 0 is a good option for applications that need high I/O throughput. However, if these applications require high availability during drive failures, RAID 0 does not provide data protection and availability.
RAID 1 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 compute system. 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 not a constraint.
Most data centers require data redundancy and performance from their RAID arrays. RAID 1+0 combines the performance benefits of RAID 0 with the redundancy benefits of RAID 1. It uses mirroring and striping techniques and combines their benefits. This RAID type requires an even number of disks, the minimum being four.

RAID 1+0 is also known as RAID 10 (Ten) or RAID 1/0. 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 storage system 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.
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.
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.
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 module) 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.
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.

This slide illustrates a single write operation on RAID 5 that contains a group of five disks. The parity (P) at the controller is calculated as follows:

\[ C_p = C_1 + C_2 + C_3 + C_4 \text{ (XOR operations)} \]

Whenever the controller performs a write I/O, parity must be computed by reading the old parity \( C_p \text{ old} \) and the old data \( C_4 \text{ old} \) from the disk, which means two read I/Os. Then, the new parity \( C_p \text{ new} \) is computed as follows:

\[ C_p \text{ new} = C_p \text{ old} - C_4 \text{ old} + C_4 \text{ new} \text{ (XOR operations)} \]

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 the 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.
The table on the slide compares different RAID levels.

<table>
<thead>
<tr>
<th>RAID level</th>
<th>Min disks</th>
<th>Available storage capacity (%)</th>
<th>Write penalty</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>50</td>
<td>2</td>
<td>Mirror</td>
</tr>
<tr>
<td>1+0</td>
<td>4</td>
<td>50</td>
<td>2</td>
<td>Mirror</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$[(n-1)/n] \times 100$</td>
<td>4</td>
<td>Parity (Supports single disk failure)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>$[(n-1)/n] \times 100$</td>
<td>4</td>
<td>Parity (Supports single disk failure)</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>$[(n-2)/n] \times 100$</td>
<td>6</td>
<td>Parity (Supports two disk failures)</td>
</tr>
</tbody>
</table>
A *hot sparing* refers to a process that temporarily replaces a failed disk drive with a spare drive in a RAID array by taking the identity of the failed disk drive. With the hot spare, one of the following methods of data recovery is performed depending on the RAID implementation:

- If parity RAID is used, the data is rebuilt onto the hot spare from the parity and the data on the surviving disk drives in the RAID set.
- If mirroring is used, the data from the surviving mirror is used to copy the data onto the hot spare.

When a new disk drive is added to the system, data from the hot spare is copied to it. The hot spare returns to its idle state, ready to replace the next failed drive. Alternatively, the hot spare replaces the failed disk drive permanently. This means that it is no longer a hot spare, and a new hot spare must be configured on the storage system.

A hot spare should be large enough to accommodate data from a failed drive. Some systems implement multiple hot spares to improve data availability.

A hot spare can be configured as automatic or user initiated, which specifies how it will be used in the event of disk failure. In an automatic configuration, when the recoverable error rates for a disk exceed a predetermined threshold, the disk subsystem tries to copy data from the failing disk to the hot spare automatically. If this task is completed before the damaged disk fails, the subsystem switches to the hot spare and marks the failing disk as unusable. Otherwise, it uses parity or the mirrored disk to recover the data. In the case of a user-initiated configuration, the administrator has control of the rebuild process. For example, the rebuild could occur overnight to prevent any degradation of system performance. However, the system is at risk of data loss if another disk failure occurs.
Lesson 3: Summary

During this lesson the following topics were covered:

- RAID implementation methods
- Three RAID techniques
- Commonly used RAID levels
- Impact of RAID on performance
- Compare RAID levels

This lesson covered RAID and its use to improve performance and protection. It covered various RAID implementations, techniques, and levels commonly used. This lesson also covered the impact of RAID on performance and compared the commonly used RAID levels.
Lesson 4: Types of Intelligent Storage Systems

This lesson covers the following topics:

- Data access methods
- Types of intelligent storage systems
- Scale-up and scale-out architectures

This lesson covers different types of data access methods. It also covers types of intelligent storage systems. Finally, this lesson covers the scale-up and scale-out architectures.
Data is accessed and stored by applications using the underlying infrastructure. The key components of this infrastructure are the OS (or file system), connectivity, and storage. The compute system controller card accesses the storage devices using predefined protocols, such as IDE/ATA, SCSI, or Fibre Channel (FC). IDE/ATA and SCSI are popularly used in small and personal computing environments for accessing internal storage. FC and iSCSI protocols are used for accessing data from an external storage device (or subsystems). External storage devices can be connected to the compute system directly or through the storage network. When the storage is connected directly to the compute system, it is referred as Direct-Attached Storage (DAS).

Data can be accessed over a network in one of the following ways: block level, file level, or object level. In general, the application requests data from the file system (or operating system) by specifying the filename and location. The file system has two components: user component and storage component. The user component of the file system performs functions such as hierarchy management, naming, and user access control. The storage component maps the files to the physical location on the storage device. The file system maps the file attributes to the logical block address of the data and sends the request to the storage device. The storage device converts the logical block address (LBA) to a cylinder-head-sector (CHS) address and fetches the data.

In a block-level access, the file system is created on a compute system, and data is accessed on a network at the block level. In this case, raw disks or logical volumes are assigned to the compute system for creating the file system.

In a file-level access, the file system is created on a separate file server or at the storage side, and the file-level request is sent over a network. Because data is accessed at the file level, this method has higher overhead, as compared to the data accessed at the block level.

Object-level access is an intelligent evolution, whereby data is accessed over a network in terms of self-contained objects with a unique object identifier. In this type of access, the file system’s user component resides on the compute system and the storage component resides on the storage system.
Types of Intelligent Storage Systems

- Block-based storage systems
- File-based storage systems
- Object-based storage systems
- Unified storage systems

Based on the type of data access, a storage system can be classified as block-based storage system, file-based storage system, object-based storage system, and unified storage system. A unified storage system provides block-based, file-based, and object-based data access in a single system.

Details on block-based, file-based, object-based, and unified storage systems are covered in the following modules.
An intelligent storage system may be built either based on scale-up or scale-out architecture.

A scale-up storage architecture provides the capability to scale the capacity and performance of a single storage system based on requirements. Scaling up a storage system involves upgrading or adding controllers and storage. These systems have a fixed capacity ceiling, which limits their scalability and the performance also starts degrading when reaching the capacity limit.

A scale-out storage architecture provides the capability to maximize its capacity by simply adding nodes to the cluster. Nodes can be added quickly to the cluster, when more performance and capacity is needed, without causing any downtime. This provides the flexibility to use many nodes of moderate performance and availability characteristics to produce a total system that has better aggregate performance and availability. Scale-out architecture pools the resources in the cluster and distributes the workload across all the nodes. This results in linear performance improvements as more nodes are added to the cluster.
Lesson 4: Summary

During this lesson the following topics were covered:

- Data access methods
- Types of intelligent storage systems
- Scale-up and scale-out architectures

This lesson covered different types of data access methods. This lesson also covered the types of intelligent storage systems. Finally, the lesson covered the scale-up and scale-out architectures.
Module 4: Summary

Key points covered in this module:

- Key components of an intelligent storage system
- HDD and SSD components, addressing, and performance
- RAID, its techniques, and its levels
- Types of intelligent storage systems

This module covered the key components of an intelligent storage system. This module also covered storage subsystems and the components, addressing, and performance parameters of hard disk drive and solid state drive. It also covered RAID, various RAID implementations, techniques, and commonly-used RAID levels. Further, this module covered the impact of RAID on performance, and compared the commonly-used RAID levels. Finally, this module covered the types of intelligent storage systems and their architectures.

This concludes the training.