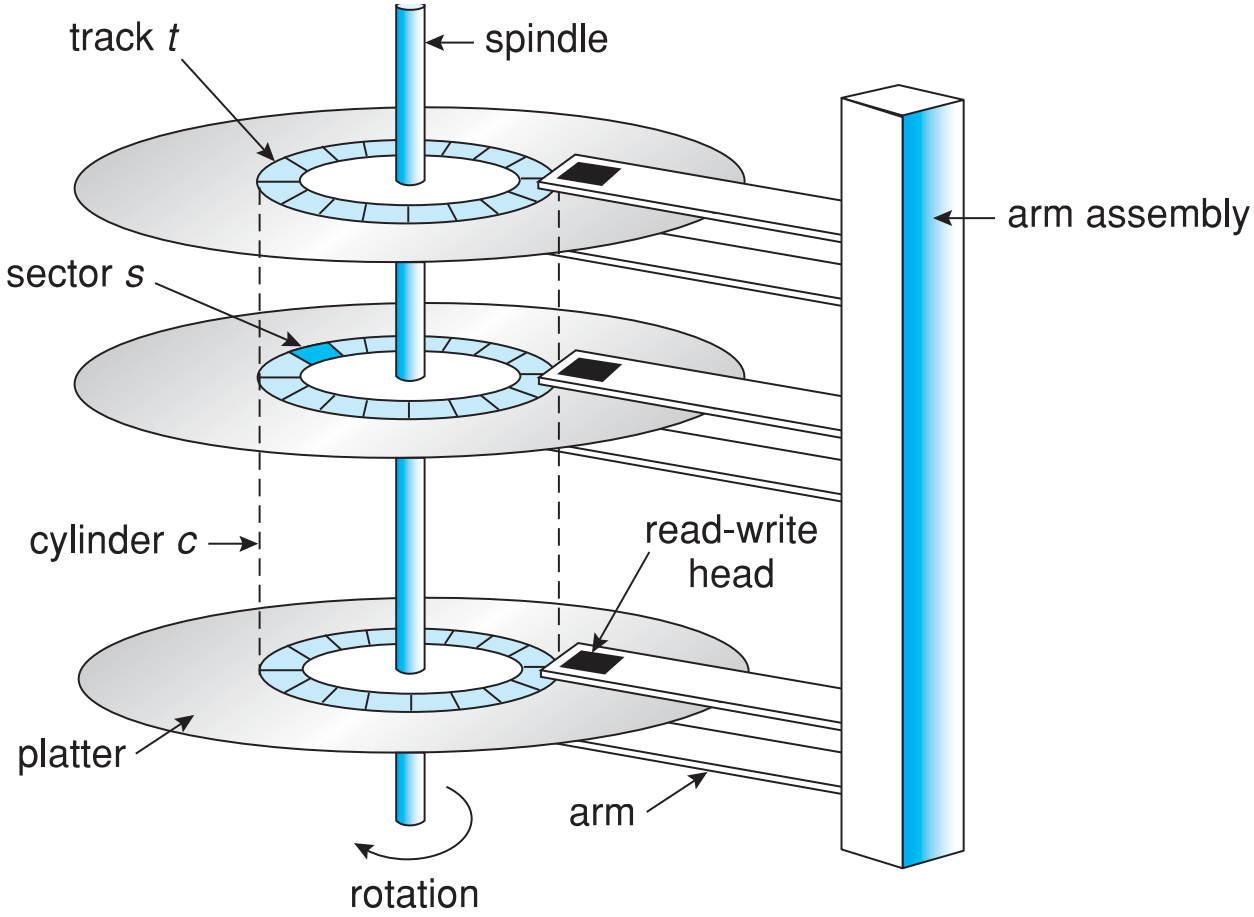
Mass Storage Structures

UNIT-IV

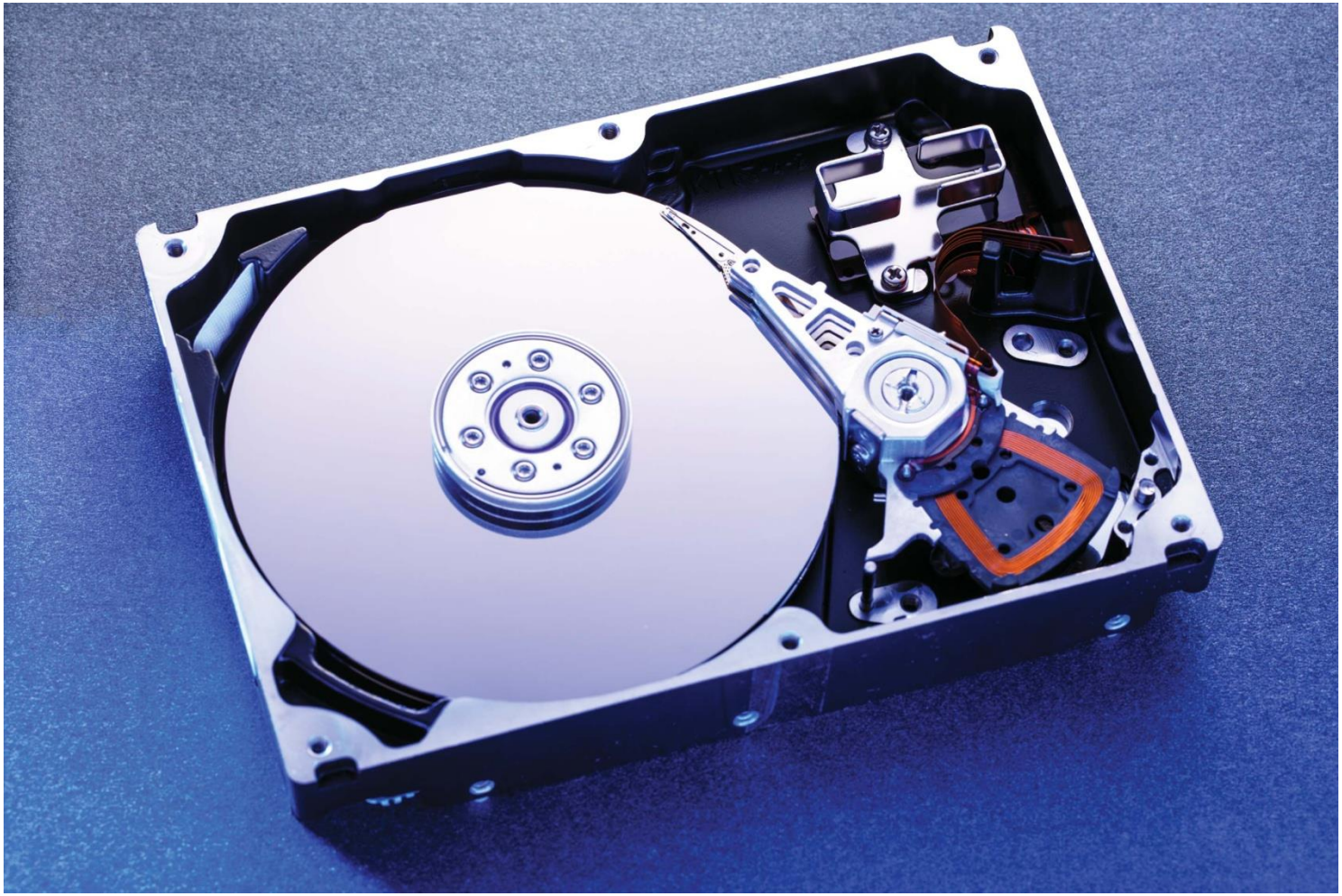
Introduction

- The main mass-storage system in modern computers is secondary storage,
  - Which is usually provided by hard disk drives (HDD) and non-volatile memory (NVM) devices
  - Some systems also have slower, larger, tertiary storage, generally consisting of magnetic tape, optical disks, or even cloud storage
- Figures in the next two slides show the moving head mechanism and hard disk

Moving Head Disk Mechanism



Hard disk



Storage Device Management- Drive Formatting, Partitions, and Volumes

- A new storage device is a blank slate
- It is just a platter of a magnetic recording material or a set of uninitialized semiconductor storage cells
- Before a storage device can store data, it must be divided into sectors that the controller can read and write
- NVM pages must be initialized and the FTL (Flash Translation Layer) must be created

- This process is called low-level formatting or physical formatting
- Most drives are low-level-formatted at the factory as a part of the manufacturing process
- This formatting enables the manufacturer
  - to test the device
  - to initialize the mapping from logical block numbers to defect-free sectors or pages on the media
- Before it can use a drive to hold files, the operating system still needs to record its own data structures on the device

- It does so in three steps

1. The first step is to ***partition*** the device into one or more groups of blocks or pages
   - The operating system can treat each partition as though it were a separate device
   - For instance, one partition can hold a file system containing a copy of the operating system's executable code, another the swap space, and another a file system containing the user files
   - Some operating systems and file systems perform the partitioning automatically when an entire device is to be managed by the file system

- In Linux, the fdisk command is used to manage partitions on storage devices
- The device, when recognized by the operating system, has its partition information read, and the operating system then creates device entries for the partitions (in /dev in Linux)
- From there, a configuration file, such as /etc/fstab, tells the operating system to mount each partition containing a file system at a specified location and to use mount options such as read-only

- **Mounting** a file system is making the file system available for use by the system and its users

2. The second step is volume creation and management
   - Sometimes, this step is implicit, as when a file system is placed directly within a partition
   - That volume is then ready to be mounted and used
   - At other times, volume creation and management is explicit
     - For example when multiple partitions or devices will be used together as a RAID set with one or more file systems spread across the devices
   - ZPS and lvm2 provide these features

3. The third step is logical formatting, or creation of a file system
   - In this step, the operating system stores the initial file-system data structures on to the device
   - These data structures may include maps of free and allocated space and an initial empty directory

- To increase efficiency, most file systems group blocks together into larger chunks, frequently called clusters

- Some operating systems give special programs the ability to use a partition as a large sequential array of logical blocks, without any file-system data structures
- This array is sometimes called the raw disk, and I/O to this array is termed raw I/O

Boot Block

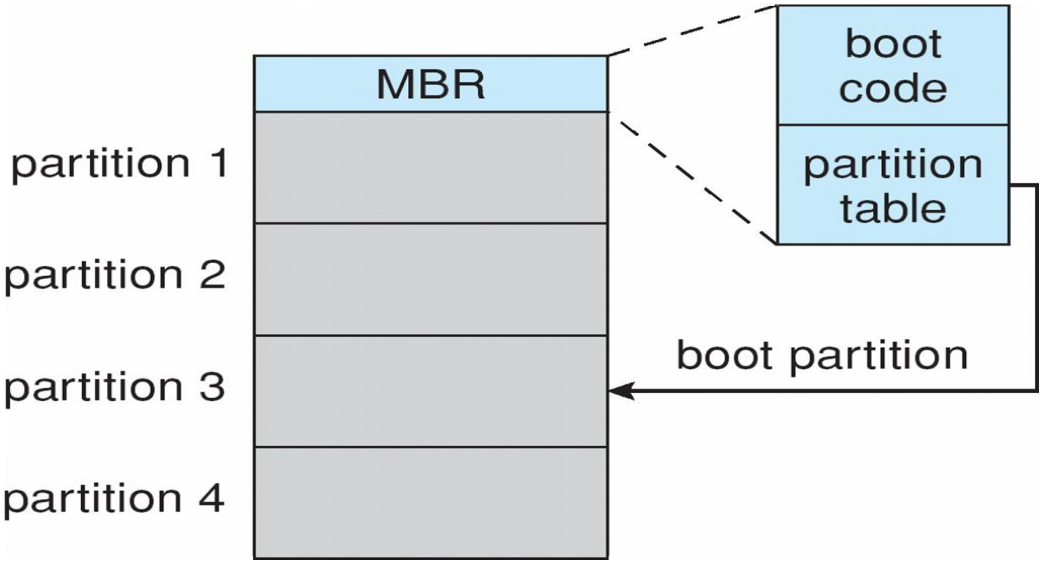
- For a computer to start running for instance, when it is powered up or rebooted—it must have an initial program to run
- This initial bootstrap loader tends to be simple
- For most computers, the bootstrap is stored in NVM flash memory firmware on the system motherboard and mapped to a known memory location
- It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory

- This tiny bootstrap loader program is also smart enough to bring in a full bootstrap program from secondary storage
- The full bootstrap program is stored in the “boot blocks” at a fixed location on the device
- The default Linux bootstrap loader is grub2
- A device that has a boot partition is called a boot disk or system disk

- The code in the bootstrap NVM instructs the storage controller to read the boot blocks into memory (no device drivers are loaded at this point) and then starts executing that code
- The full bootstrap program is more sophisticated than the bootstrap loader
  - it is able to load the entire operating system from a non-fixed location on the device and to start the operating system running

- Let’s consider as an example the boot process in Windows
  - First, Windows allows a drive to be divided into partitions, and one partition identified as the boot partition contains the operating system and device drivers
  - The Windows system places its boot code in the first logical block on the hard disk or first page of the NVM device, which it terms the master boot record, or MBR
  - Booting begins by running code that is resident in the system’s firmware
  - This code directs the system to read the boot code from the MBR, understanding just enough about the storage controller and storage device to load a sector from it

- In addition to containing boot code, the MBR contains a table listing the partitions for the drive and a flag indicating which partition the system is to be booted from, as illustrated in Figure below



- Once the system identifies the boot partition, it reads the first sector/page from that partition (called the boot sector), which directs it to the kernel
- It then continues with the remainder of the boot process, which includes loading the various subsystems and system services

Bad Blocks

- Because disks have moving parts and small tolerances they are prone to failure
- Sometimes the failure is complete
  - In this case, the disk needs to be replaced and its contents restored from backup media to the new disk
- More frequently, one or more sectors become defective
- Most disks even come from the factory with bad blocks

- Depending on the disk and controller in use, these blocks are handled in a variety of ways
- On older disks, such as some disks with IDE controllers, bad blocks are handled manually
  - One strategy is to scan the disk to find bad blocks while the disk is being formatted
  - Any bad blocks that are discovered are flagged as unusable so that the file system does not allocate them
  - If blocks go bad during normal operation, a special program must be run manually to search for the bad blocks and to lock them away

  - Data that resided on the bad blocks usually are lost
- More sophisticated disks are smarter about bad-block recovery
- They use the scheme known as sector sparing or forwarding
  - The controller maintains a list of bad blocks on the disk
  - The list is initialized during the low-level formatting at the factory and is updated over the life of the disk
  - Low-level formatting also sets aside spare sectors not visible to the operating system
  - The controller can be told to replace each bad sector logically with one of the spare sectors

- A typical bad-sector transaction might be as follows
  - The operating system tries to read logical block 87
  - The controller calculates the ECC and finds that the sector is bad
  - It reports this finding to the operating system as an I/O error
  - The device controller replaces the bad sector with a spare
  - After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller
- Sector slipping
  - Is an alternative to sector sparing
  - some controllers can be instructed to replace a bad block by sector slipping

- Example
  - Suppose that logical block 17 becomes defective and the first available spare follows sector 202
  - Sector slipping then remaps all the sectors from 17 to 202, moving them all down one spot
  - That is, sector202 is copied into the spare, then sector 201 into 202, then 200 into 201, and so on, until sector 18 is copied into sector 19
  - Slipping the sectors in this way frees up the space of sector 18 so that sector 17 can be mapped to it

- Recoverable soft errors may trigger a device activity in which a copy of the block data is made and the block is spared or slipped
- An unrecoverable hard error, however, results in lost data
- NVM devices also have bits, bytes, and even pages that either are non functional at manufacturing time or go bad over time
- Management of those faulty areas is simpler than for HDDs because there is no seek time performance loss to be avoided

Swap Space Management

- Swap-space management is another low-level task of the operating system
- Virtual memory uses secondary storage space as an extension of main memory
- Since drive access is much slower than memory access, using swap space significantly decreases system performance
- The main goal for the design and implementation of swap space is to provide the best throughput for the virtual memory system

Swap space use

- Swap space is used in various ways by different operating systems, depending on the memory-management algorithms in use
  - For instance, systems that implement swapping may use swap space to hold an entire process image, including the code and data segments
  - Paging systems may simply store pages that have been pushed out of main memory
- The amount of swap space needed on a system can therefore vary from a few megabytes of disk space to gigabytes

- The amount of swap space needed depends on
  - the amount of physical memory ,
  - the amount of virtual memory it is backing
  - the way in which the virtual memory is used
- It is safer to overestimate than to underestimate the amount of swap space required, because
  - if a system runs out of swap space it may be forced to abort processes or may crash entirely
- Overestimation wastes secondary storage space that could otherwise be used for files, but it does no other harm

- Some systems recommend the amount to be set aside for swap space
  - E.g. Solaris
- Most Linux systems use considerably less swap space
- Some operating systems—including Linux—allow the use of multiple swap spaces, including both files and dedicated swap partitions

Swap space location

- A swap space can reside in one of two places
  - It can be carved out of the normal file system
  - It can be in a separate partition
- If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, name it, and allocate its space
- Swap space can be created in a separate raw partition
  - No file system or directory structure is placed in this space

- A separate swap-space storage manager is used to allocate and deallocate the blocks from the raw partition
- This manager uses algorithms optimized for speed rather than for storage efficiency, because swap space is accessed much more frequently than file systems when it is used
- Internal fragmentation may increase, but this tradeoff is acceptable
  - because the life of data in the swap space generally is much shorter than that of files in the file system

- Since swap space is reinitialized at boot time, any fragmentation is short-lived
- The raw-partition approach creates a fixed amount of swap space during disk partitioning
- Adding more swap space requires
  - either repartitioning the device
  - adding another swap space elsewhere
- Some operating systems are flexible and can swap both in raw partitions and in file-system space E.g: Linux

Swap-Space Management: An Example

- Use of swap space can be illustrated by following the evolution of swapping and paging in various UNIX systems
- The traditional UNIX kernel started with an implementation of swapping that copied entire processes between contiguous disk regions and memory
- UNIX later evolved to a combination of swapping and paging as paging hardware became available

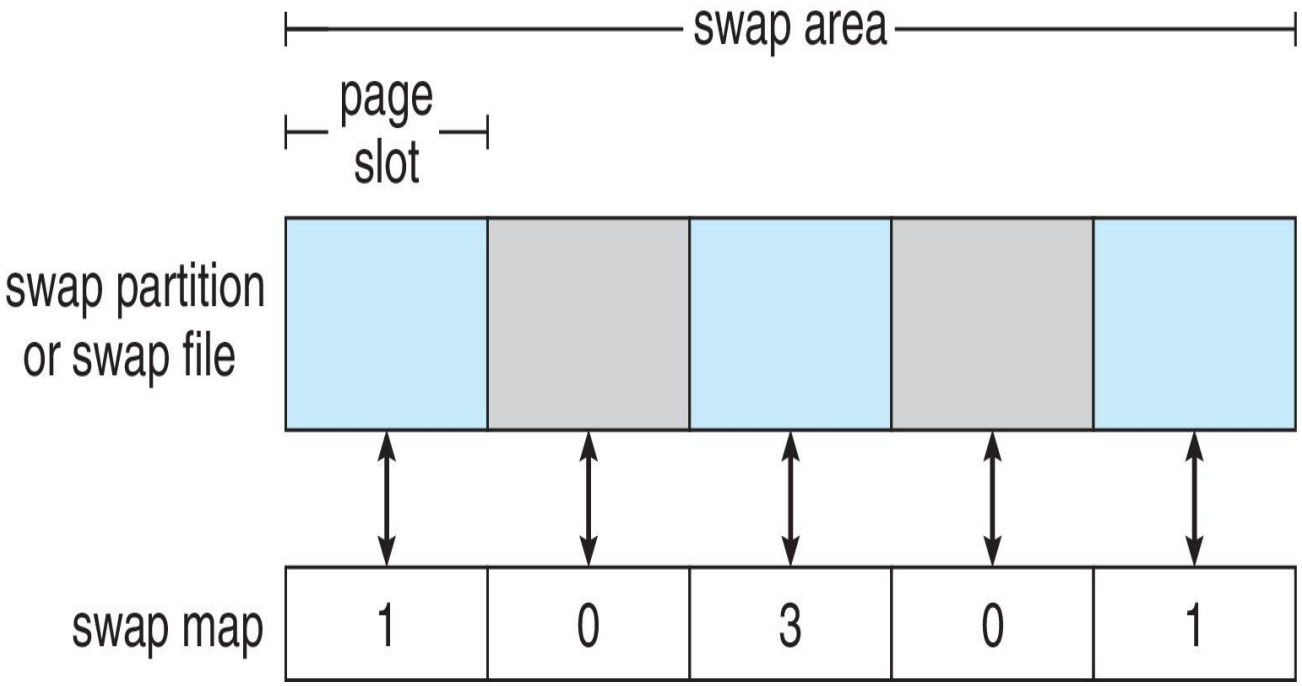
- In Solaris 1 (SunOS), the designers changed standard UNIX methods to improve efficiency and reflect technological developments
- When a process executes, text-segment pages containing code are brought in from the file system, accessed in main memory, and thrown away if selected for pageout
- It is more efficient to reread a page from the file system than to write it to swap space and then reread it from there
- Swap space is only used as a backing store for pages of anonymous memory (memory not backed by any file)

- Anonymous memory includes memory allocated for the stack, heap and uninitialized data of a process
- More changes were made in later versions of Solaris
  - The biggest change is that Solaris now allocates swap space only when a page is forced out of physical memory, rather than when the virtual memory page is first created
  - This scheme gives better performance on modern computers, which have more physical memory than older systems and tend to page less

- Linux is similar to Solaris in that swap space is now used only for anonymous memory
- Linux allows one or more swap areas to be established
- A swap area may be in either a swap file on a regular file system or a dedicated swap partition
- Each swap area consists of a series of 4-KB page slots, which are used to hold swapped pages
- Associated with each swap area is a swap map—an array of integer counters, each corresponding to a page slot in the swap area

- If the value of a counter is 0, the corresponding page slot is available
- Values greater than 0 indicate that the page slot is occupied by a swapped page
- The value of the counter indicates the number of mappings to the swapped page
- For example, a value of 3 indicates that the swapped page is mapped to three different processes
  - which can occur if the swapped page is storing a region of memory shared by three processes

- The data structures for swapping on Linux systems are shown in Figure below:



Storage attachment

- Computers access secondary storage in three ways:
  1. Host-attached storage
  2. Network-attached storage
  3. Cloud storage

Host-Attached Storage

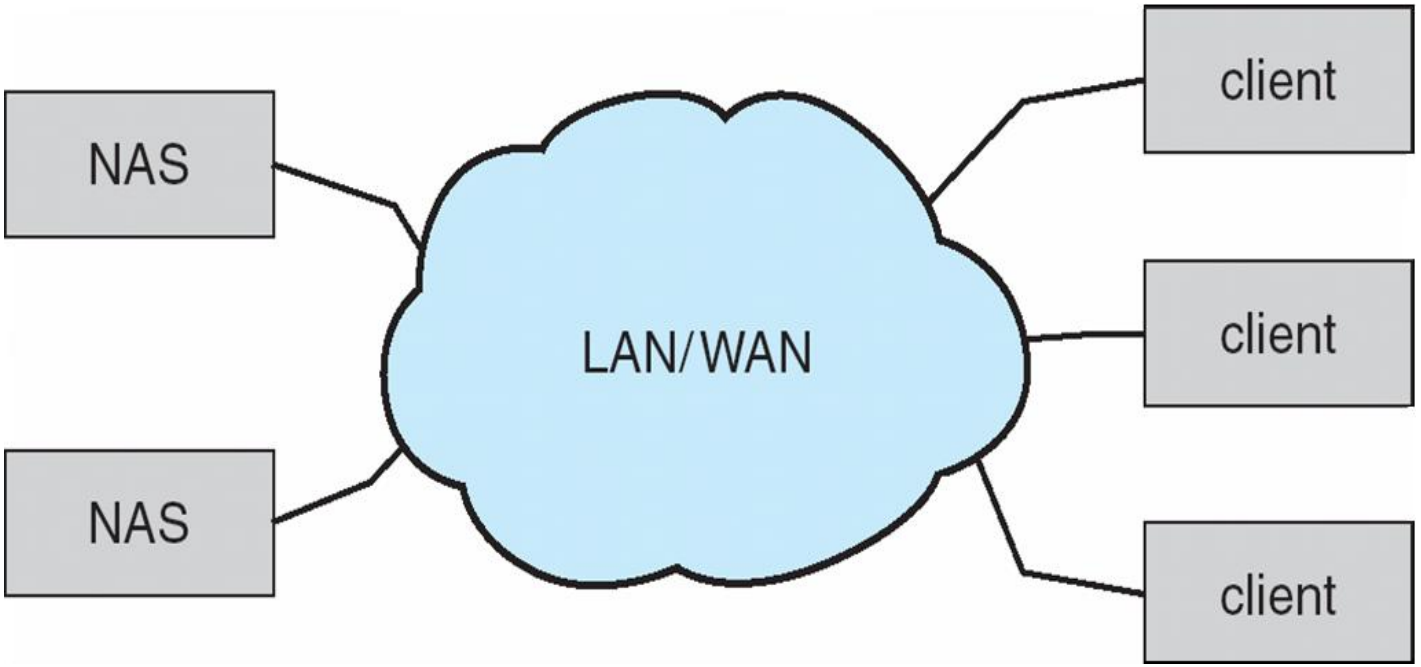
- Host-attached storage is storage accessed through local I/O ports
- These ports use several technologies, the most common being SATA
- A typical system has one or a few SATA (Serial Advanced Technology Attachment) ports

- To allow a system to gain access to more storage
  - either an individual storage device, a device in a chassis, or multiple drives in a chassis can be connected via USB FireWire or Thunderbolt ports and cables
- High-end workstations and servers generally need more storage or need to share storage, so use more sophisticated I/O architectures such as fibre channel(FC), a high-speed serial architecture that can operate over optical fibre or over a four-conductor copper cable

- A wide variety of storage devices are suitable for use as host-attached storage. They are :
  - HDDs
  - NVM devices
  - CD, DVD, Blu-ray, and tape drives
  - Storage-Area Networks (SANs)
- The I/O commands that initiate data transfers to a host-attached storage device are reads and writes of logical data blocks directed to specifically identified storage units

Network-Attached Storage

- Network-attached storage (NAS) (Figure below) provides access to storage across a network



- A NAS device can be either
  - a special-purpose storage system or
  - a general computer system that provides its storage to other hosts across the network
- Clients access network-attached storage via a remote-procedure call interface such as NFS (Network File System) for UNIX and Linux systems or CIFS (Common Internet File System) for Windows machines

- The remote procedure calls (RPCs) are carried via TCP or UDP over an IP (Internet Protocol) network
  - Uses the same local-area network (LAN) that carries all data traffic to the clients
- The network-attached storage unit is usually implemented as a storage array with software that implements the RPC interface
- CIFS and NFS provide various locking features, allowing the sharing of files between hosts accessing a NAS with those protocols

- For example, a user logged into multiple NAS clients can access her home directory from all of those clients simultaneously
- Network-attached storage provides a convenient way for all the computers on a LAN to share a pool of storage with the same ease of naming and access enjoyed with local host-attached storage
- iSCSI is the latest network-attached storage protocol
  - In essence, it uses the IP network protocol to carry the SCSI protocol

- Thus, networks rather than SCSI cables can be used as the interconnects between hosts and their storage
- As a result, hosts can treat their storage as if it were directly attached, even if the storage is distant from the host
- NFS and CIFS present a file system and send parts of files across the network, iSCSI sends logical blocks across the network and leaves it to the client to use the blocks directly or create a file system with them

Cloud Storage

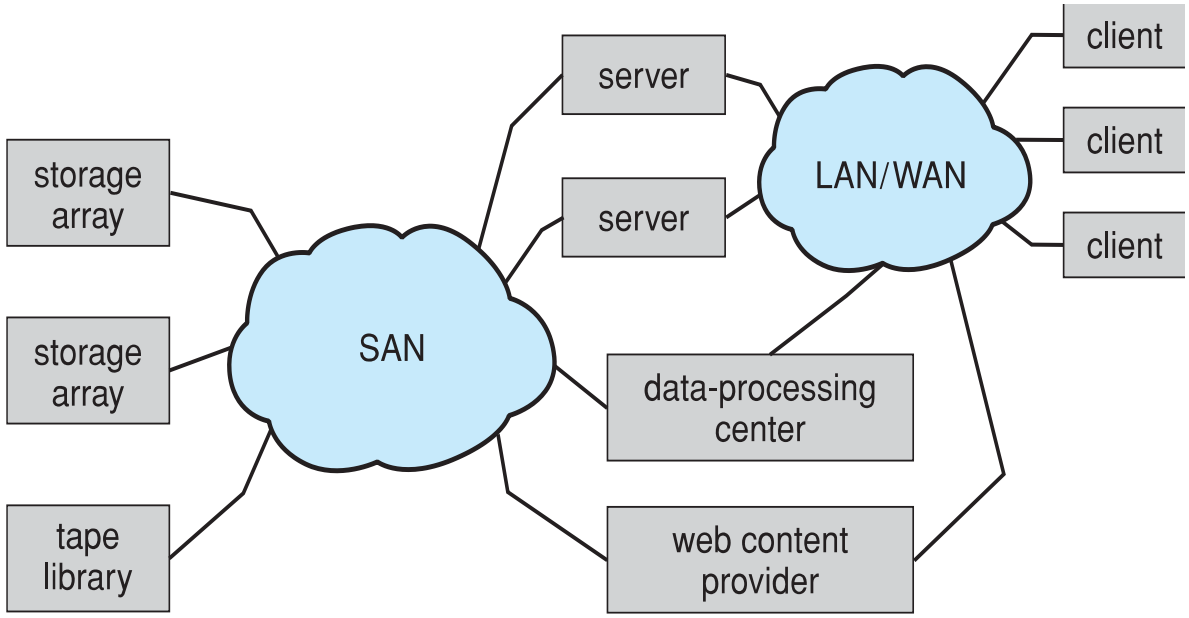
- Cloud storage provides access to storage across a network
- The storage is accessed over the Internet or another WAN to a remote data center that provides storage for a fee or even for free
- Cloud storage is API based, and programs use the APIs to access the storage
- Amazon S3 is a leading cloud storage offering

- Dropbox is an example of a company that provides apps to connect to the cloud storage that it provides
- Other examples include Microsoft OneDrive and Apple iCloud
- One reason that APIs are used instead of existing protocols is the latency and failure scenarios of a WAN
  - NAS protocols were designed for use in LANs, which have lower latency than WANs and are much less likely to lose connectivity between the storage user and the storage device
  - If a LAN connection fails, a system using NFS or CIFS might hang until it recovers

- With cloud storage, failures like that are more likely, so an application simply pauses access until connectivity is restored

Storage Area Network and Storage Arrays

- A storage-area network (SAN) is a private network (using storage protocols rather than networking protocols) connecting servers and storage units as shown in Figure below:



- The power of a SAN lies in its flexibility
- Multiple hosts and multiple storage arrays can attach to the same SAN, and storage can be dynamically allocated to hosts
- A SAN switch allows or prohibits access between the hosts and the storage
  - As one example, if a host is running low on disk space, the SAN can be configured to allocate more storage to that host
- SANs make it possible
  - for clusters of servers to share the same storage
  - for storage arrays to include multiple direct host connections

- SANs typically have more ports and cost more than storage arrays
- FC (Fibre channel) is the most common SAN interconnect
- Another SAN interconnect is InfiniBan (IB)
  - A special purpose bus architecture that provides hardware and software support for high-speed interconnection networks for servers and storage units

- A storage array
  - is a purpose-built device (see Figure in the next slide) that includes SAN ports, network ports or both
  - It also contains drives to store data
  - a controller or redundant set of controllers to manage the storage and allow access to the storage across the networks
  - The controllers are composed of CPUs, memory, software that implement the features of the array which can include network protocols RAID protection, snapshots, replication, compression, deduplication, and encryption

A Storage Array