Chapter 10.C: Mass-Storage Systems

Some slides from the Silberschatz, Galvin and Gagne, "Operating System Concepts", 10th eds, 2018.

Outline

- Overview of Mass Storage Structure
- HDD Scheduling
- NVM Scheduling
- Error Detection and Correction
- Storage Device Management
- Swap-Space Management
- Storage Attachment
- RAID Structure

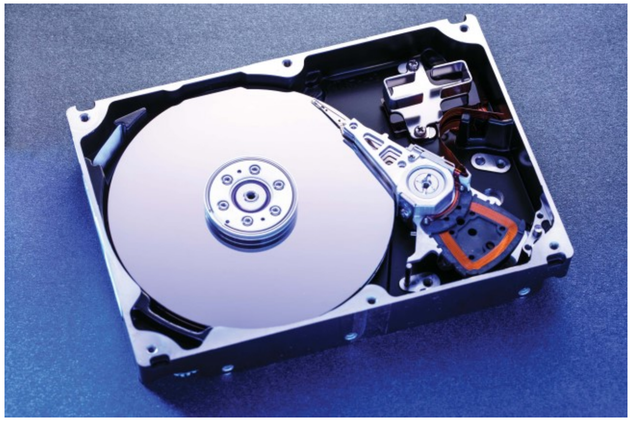
Objectives

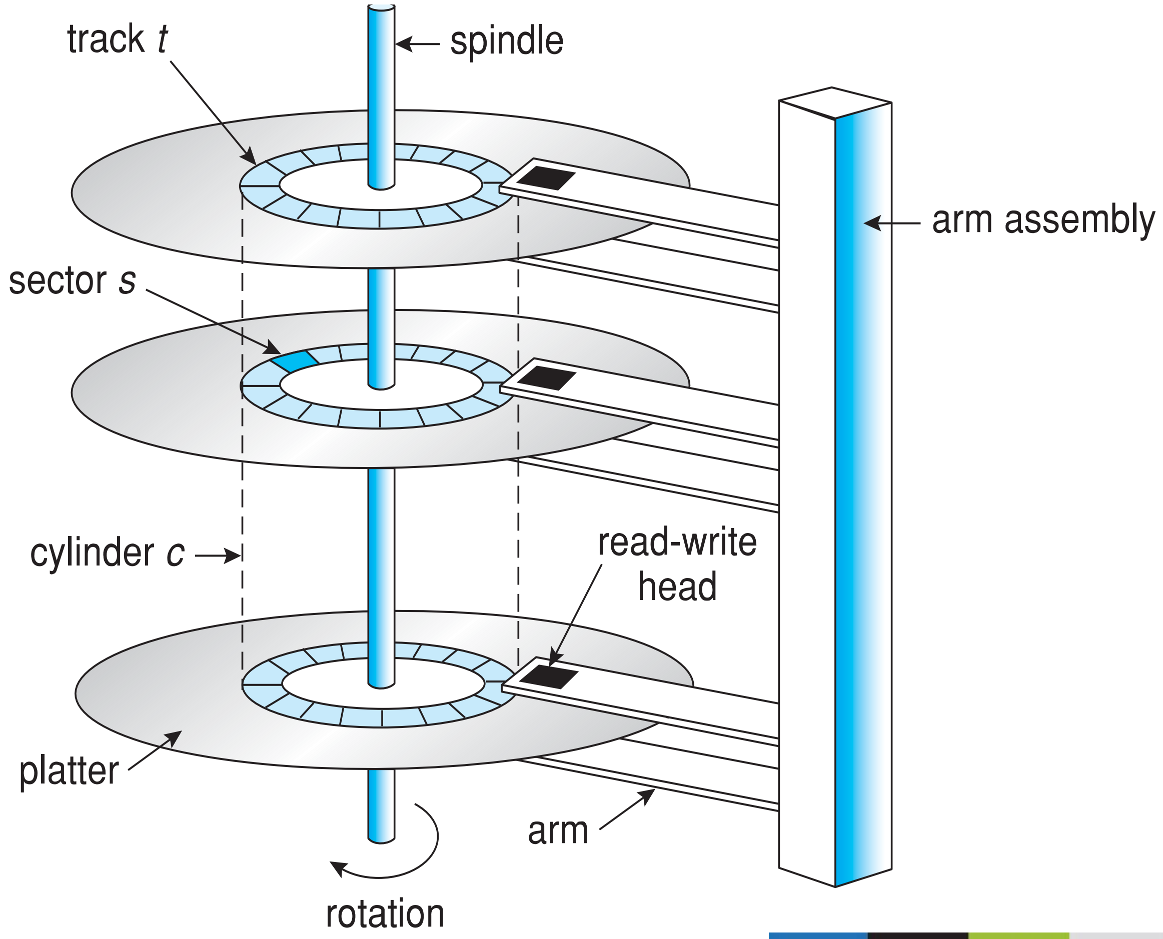
- Describe the *physical structure of secondary storage devices* and the effect of a device's structure on its uses
- Explain the *performance characteristics* of mass-storage devices
- Evaluate *I/O scheduling algorithms*
- Discuss *operating-system services* provided for mass storage, including **RAID**

Overview of Mass Storage Structure

- Bulk of secondary storage for modern computers is *hard disk drives* (**HDD**s) and *nonvolatile memory* (**NVM**) devices
- HDDs spin *platters* of magnetically-coated material under moving read-write *heads*
  - Disks can be removable
  - *Head crash* results from disk head making contact with the disk surface
    - That's bad
- Performance
  - *Drive rotation* is at *60* to *250* times per second
  - *Transfer rate* is rate at which data flow between drive and computer
  - *Positioning time* (random-access time) is time to move disk arm to desired cylinder (*seek time*) and time for desired sector to rotate under the disk head (*rotational latency*)

Moving-head Disk Mechanism





Example of Hard Disk Drives

- *Platters* range from *.85"* to *14"* (historically)
  - Commonly *3.5"*, *2.5"*, and *1.8"*
- Range from *30GB* to *3TB* per drive
- Performance
  - *Transfer Rate* – theoretical – *6 Gb/sec*
    - *Effective Transfer Rate* – real – *1Gb/sec*
  - *Seek time* from *3ms* to *12ms* (e.g., *9ms* is common for desktop drives)
    - *Average seek time* is measured or calculated based on 1/3 of tracks
  - *Latency* based on spindle speed
    - ***1 / (RPM / 60) = 60 / RPM***
    - *Average latency* = ½ latency

Hard Disk Performance

- *Access Latency* = *Average access time* = *average seek time* + *average rotational latency*
  - For fastest disk: *3ms* + *2ms* = *5ms*
  - For slow disk: *9ms* + *5.56ms* = *14.56ms*
- *Average I/O time* = *average access time* + (*amount to transfer* / *transfer rate*) + *controller overhead*
- For example, to transfer a *4KB* block on a *7200 RPM* disk with a *5ms* average seek time, *1Gb/sec* transfer rate with a *0.1ms* controller overhead, the *average I/O time for 4KB block* is
  - = *5ms* + *4.17ms* + *0.1ms* + transfer time
    - Transfer time = *4KB / 1Gb/s * 8Gb / GB * 1GB / 10242KB = 32 / (10242) = 0.031 ms*
  - = *9.27ms* + *.031ms* = *9.301ms*

Nonvolatile Memory Devices

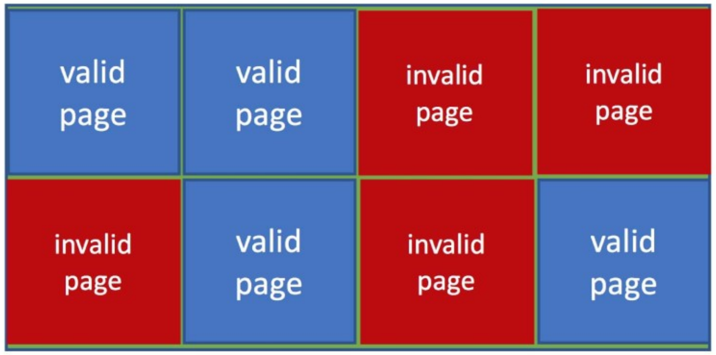
- If disk-drive like, then called *solid-state disks* (**SSD**s)
- Other forms include *USB drives* (thumb drive, flash drive), *DRAM disk* replacements, surface-mounted on motherboards, and main storage in devices like smartphones
  - Busses can be too slow –> connect directly to PCIe for example
  - No moving parts, so no seek time or rotational latency
  - Can be more reliable than HDDs
  - More expensive per MB
  - Maybe have shorter life span – need careful management
  - Less capacity
  - But much faster

Nonvolatile Memory Devices (Cont.)

- Have characteristics that present challenges
- Read and written in “page” increments (think sector) but can’t overwrite in place
  - Must first be erased, and erases happen in larger ”block” increments
  - Can only be erased a limited number of times before worn out ~ 100,000
  - Life span measured in *drive writes per day* (**DWPD**)
- E.g., A 1TB NAND drive with rating of 5DWPD is expected to have 5TB per day written within warrantee period without failing

NAND Flash Controller Algorithms

- With *no overwrite*, pages end up with *mix of valid and invalid data*
- To track which logical blocks are valid, controller maintains *flash translation layer* (**FTL**) table
- Also implements *garbage collection* (**GC**) to free invalid page space
- Allocates *overprovisioning* to provide working space for GC
- Each cell has lifespan, so *wear leveling* needed to write equally to all cells



NAND block with valid and invalid pages

Volatile Memory

- **DRAM** frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, be used like very fast secondary storage
- **RAM** *drives* (with many names, including **RAM** *disks*) present as raw block devices, commonly file system formatted
  - Found in all major operating systems
    - Linux */dev/ram*, macOS *diskutil* to create them, Solaris */tmp* of file system type *tmpfs*
  - Computers have buffering, caching via RAM, so why RAM drives?
    - Caches / buffers allocated / managed by programmer, operating system, hardware
    - *RAM drives under user control*
- Used as high speed temporary storage
  - Programs could share bulk date, quickly, by reading/writing to RAM drive

Disk Attachment

- *Host-attached storage* accessed through I/O ports talking to I/O busses. Several busses available, including *advanced technology attachment* (**ATA**), *serial ATA* (**SATA**), **eSATA**, *serial attached SCSI* (**SAS**), *universal serial bus* (**USB**), and *fibre channel* (**FC**)
  - Because NVM much faster than HDD, new fast interface for NVM called *NVM express* (**NVMe**), connecting directly to **PCI** *bus*
- Data transfers on a bus carried out by special electronic processors called *controllers* (or *host-bus adapters*, **HBA**s)
  - Host controller on the computer end of the bus, device controller on device end
  - Computer places command on host controller, using memory-mapped I/O ports
  - Host controller sends messages to device controller
  - Data transferred via DMA between device and computer DRAM

Address Mapping

- *Disk drives* are addressed as *large 1-dimensional arrays of logical blocks*, where the logical block is the smallest unit of transfer
  - *Low-level formatting* creates logical blocks on physical media
- The 1-dimensional array of logical blocks is *mapped* into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

HDD Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a *fast access time* and *disk bandwidth*
  - Seek time ≈ seek distance
  - Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer
- *Minimize seek time*

Disk Scheduling (Cont.)

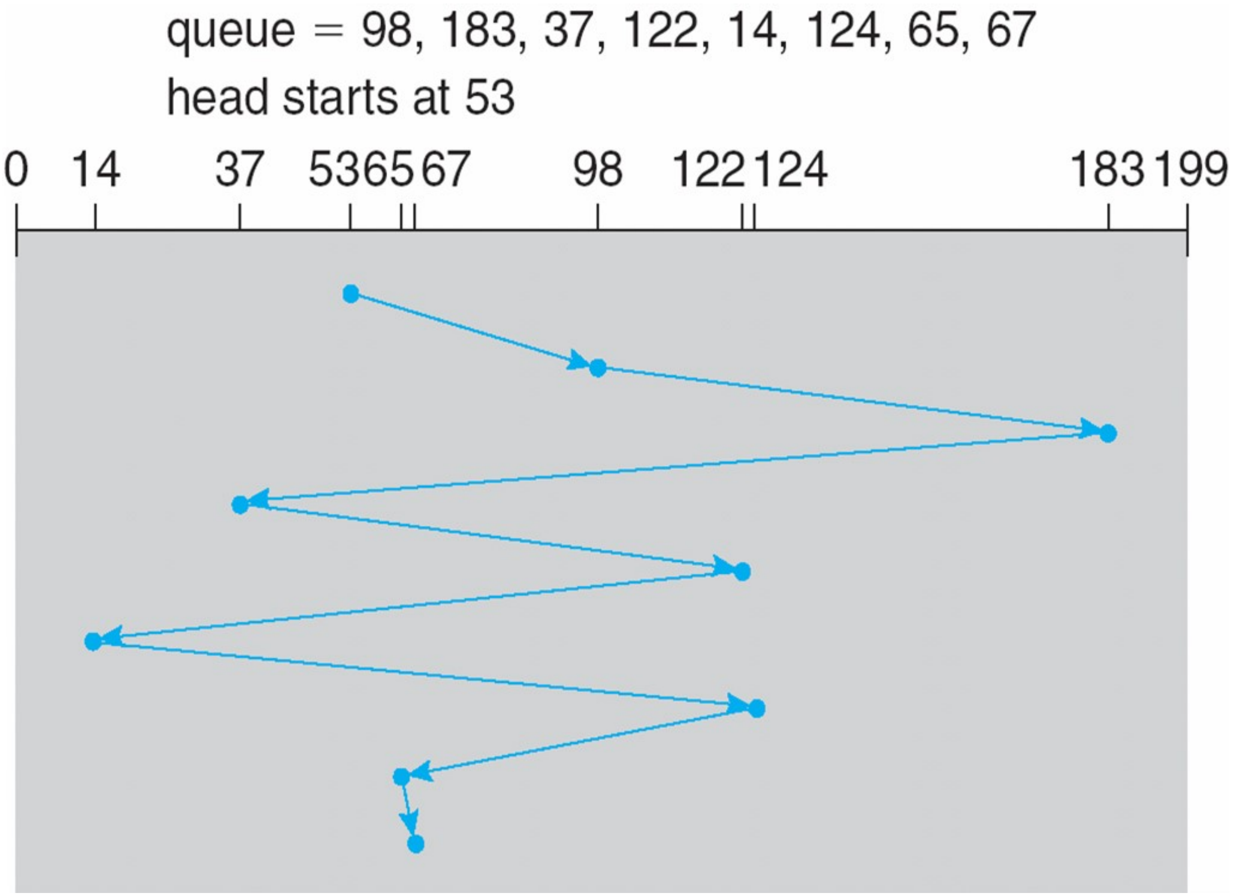
- There are many sources of *disk I/O request*
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - *Optimization algorithms* only make sense when a queue exists
  - In the past, operating system responsible for queue management, disk drive head scheduling
    - Now, built into the storage devices, controllers
  - Just provide **LBA**s, handle sorting of requests

Disk Scheduling (Cont.)

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)
- *Several algorithms exist to schedule the servicing of disk I/O requests*
- The analysis is true for one or many platters
- E.g., We illustrate scheduling algorithms with a request queue (0-199)
  - **98, 183, 37, 122, 14, 124, 65, 67**
  - Head pointer –> 53

First Come First Served (FCFS)

- Illustration shows total head movement of *640* cylinders

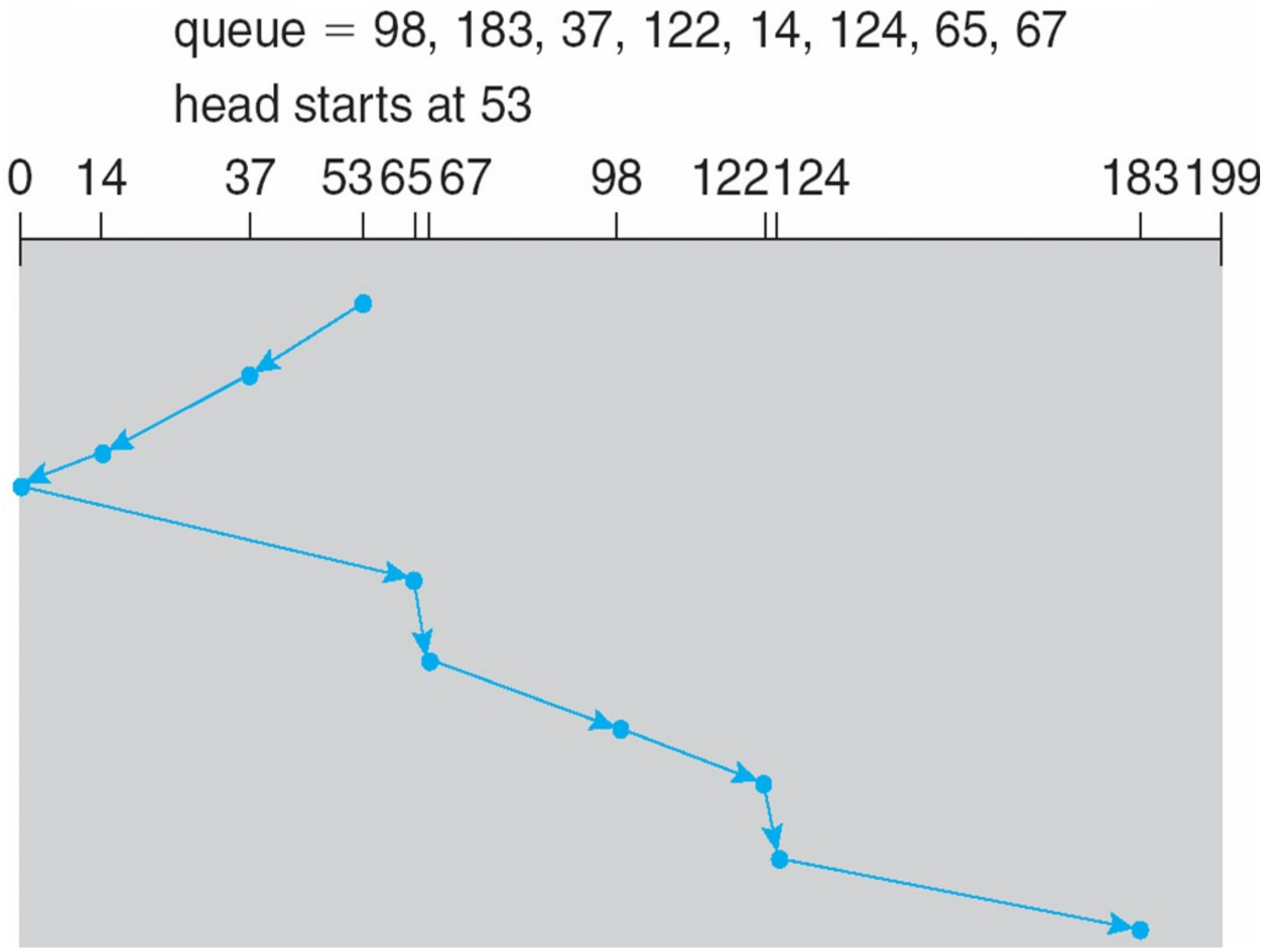


SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- SCAN algorithm sometimes called the *elevator algorithm*
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

- Illustration shows total head movement of *208* cylinders

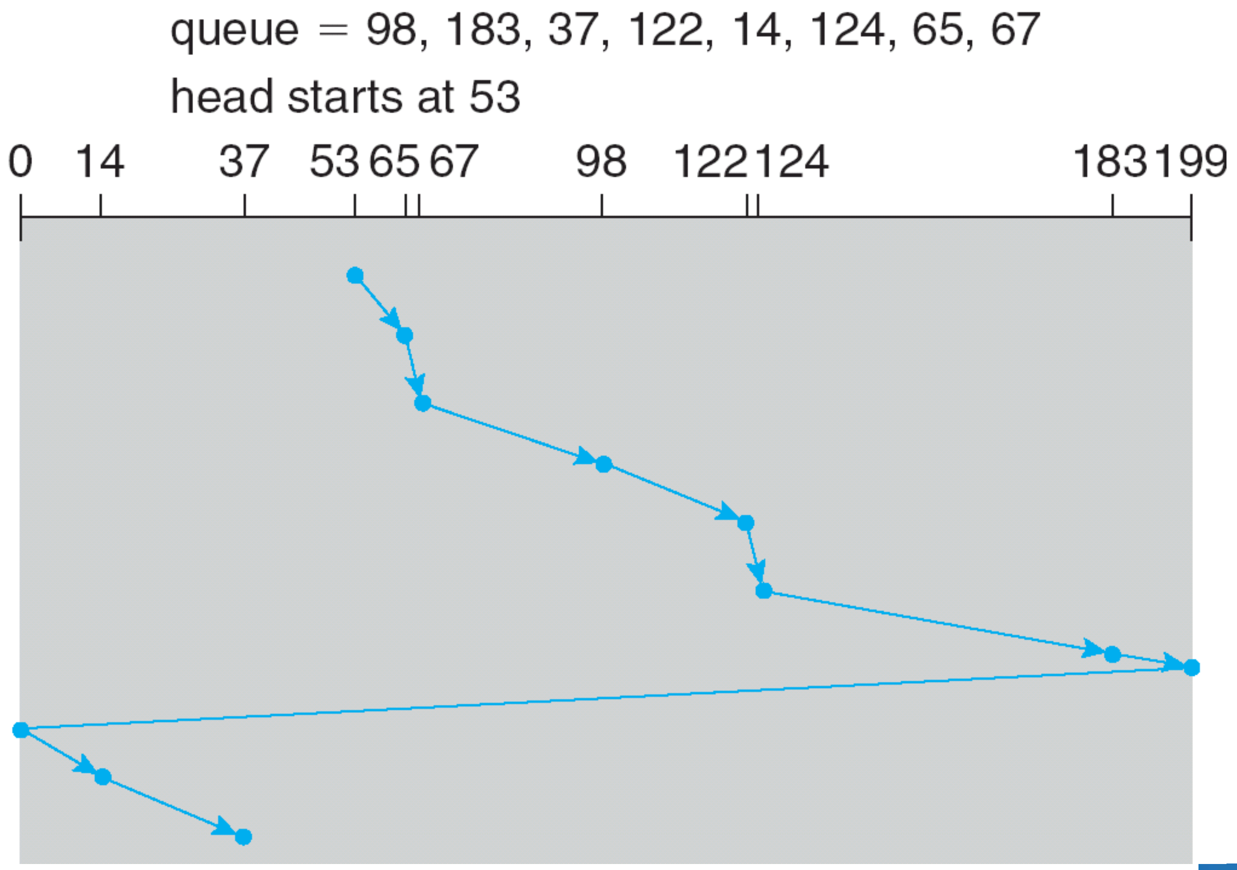


C-SCAN

- Provides a *more uniform wait time* than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, *without servicing any requests on the return trip*
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

C-SCAN (Cont.)

- Total number of cylinders?



Selecting a Disk-Scheduling Algorithm

- **FCFS** is common and has a natural appeal
- **SCAN** and **C-SCAN** perform better for systems that place a heavy load on the disk (less starvation, but still possible)
- To avoid starvation Linux implements *deadline scheduler*
  - Maintains separate read and write queues, gives read priority
    - Because processes more likely to block on read than write
  - Implements four queues: 2 x read and 2 x write
    - 1 read and 1 write queue sorted in LBA order, (implementing C-SCAN)
    - 1 read and 1 write queue sorted in FCFS order
    - All I/O requests sent in batch sorted in that queue's order
    - After each batch, checks if any requests in FCFS older than configured age (default 500ms)
      - If so, LBA queue containing that request is selected for next batch of I/O
  - In RHEL 7, **NOOP** and *completely fair queueing scheduler* (**CFQ**) are also available, defaults vary by storage device

NVM Scheduling

- *No disk heads* or *rotational latency* but still room for optimization
- In RHEL 7, **NOOP** (no scheduling) is used but *adjacent LBA requests are combined*
  - NVM best at random I/O, HDD at sequential
  - Throughput can be similar
  - Input/Output operations per second (IOPS) much higher with NVM (hundreds of thousands vs hundreds)
  - But write amplification (one write, causing garbage collection and many read/writes) can decrease the performance advantage

Error Detection and Correction

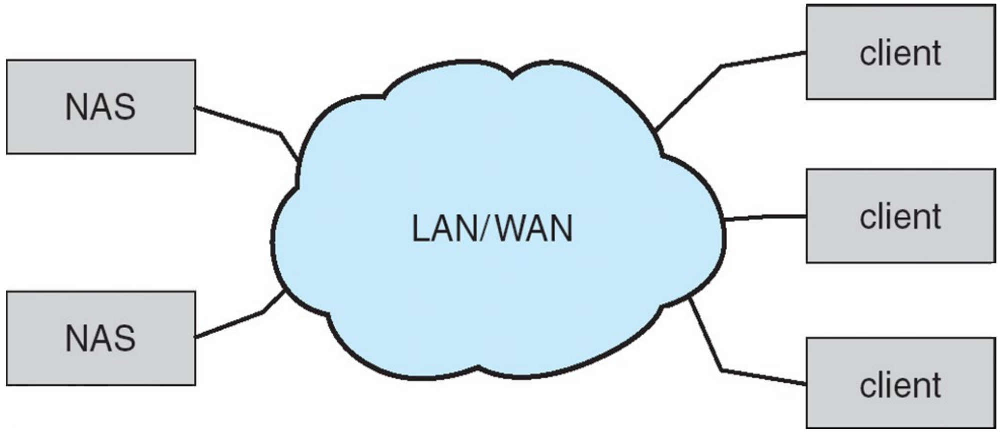
- Fundamental aspect of many parts of computing (memory, networking, storage)
- *Error detection* determines if there a problem has occurred (for example a bit flipping)
  - If detected, can halt the operation
  - Detection frequently done via parity bit
  - *Parity* – one form of checksum – uses modular arithmetic to compute, store, compare values of fixed-length words
  - Another error-detection method common in networking is *Cyclic Redundancy Check* (CRC) which uses hash function to detect multiple-bit errors
- *Error-correction code* (**ECC**) not only detects, but can correct some errors
  - *Soft errors* correctable, *hard errors* detected but not corrected

Storage Attachment

- Computers access storage in three ways
  - host-attached
  - network-attached
  - cloud
- Host attached access through local I/O ports, using one of several technologies
  - To attach many devices, use storage busses such as USB, firewire, thunderbolt
  - High-end systems use *fibre channel* (**FC**)
    - High-speed serial architecture using fibre or copper cables
    - Multiple hosts and storage devices can connect to the FC fabric

Network-Attached Storage

- *Network-attached storage* (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
  - Remotely attaching to file systems
- **NFS** and **CIFS** are common protocols
- Implemented via *remote procedure calls* (**RPC**s) between host and storage over typically TCP or UDP on IP network
- **iSCSI** protocol uses IP network to carry the SCSI protocol
  - Remotely attaching to devices (blocks)



Cloud Storage

- Similar to NAS, provides access to storage across a network
    - Unlike NAS, *accessed over the Internet or a WAN to remote data center*
- NAS presented as just another file system, while cloud storage is API based, with programs using the APIs to provide access
    - Examples include Dropbox, Amazon S3, Microsoft OneDrive, Apple iCloud
    - Use APIs because of latency and failure scenarios (NAS protocols wouldn't work well)

Storage Array

- Can just attach disks, or arrays of disks
- Avoids the NAS drawback of using network bandwidth
- Storage Array has controller(s), provides features to attached host(s)
  - Ports to connect hosts to array
  - Memory, controlling software (sometimes NVRAM, etc)
  - A few to thousands of disks
  - RAID, hot spares, hot swap (discussed later)
  - Shared storage -> more efficiency
  - Features found in some file systems
    - Snaphots, clones, thin provisioning, replication, deduplication, etc

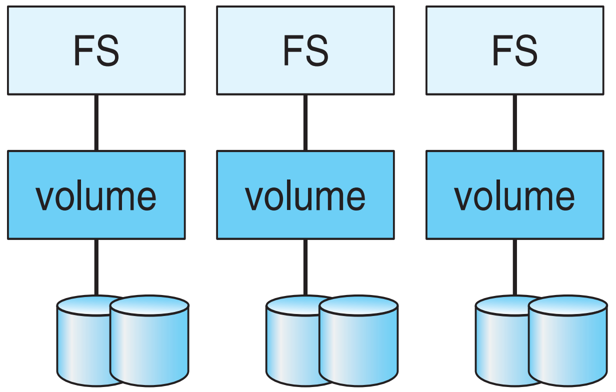
RAID Structure

- RAID – redundant array of inexpensive disks
  - multiple disk drives provides reliability via redundancy
- Increases the mean time to failure
- Mean time to repair – exposure time when another failure could cause data loss
- Mean time to data loss based on above factors
- If mirrored disks fail independently, consider disk with 1300,000 mean time to failure and 10 hour mean time to repair
  - Mean time to data loss is *100, 0002 / (2 ∗ 10) = 500 ∗ 106 hours*, or *57,000 years*!
- Frequently combined with NVRAM to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

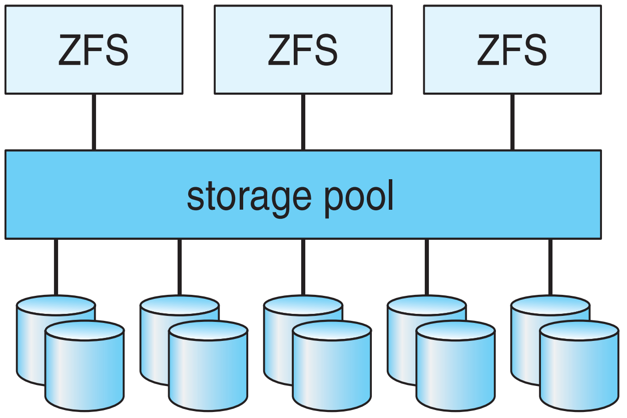
RAID (Cont.)

- Disk striping uses a group of disks as one storage unit
- RAID is arranged into six different levels
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - Mirroring or shadowing (RAID 1) keeps duplicate of each disk
  - Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provides high performance and high reliability
  - Block interleaved parity (RAID 4, 5, 6) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic replication of the data between arrays is common
- Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Object Storage

- General-purpose computing, file systems not sufficient for very large scale
- Another approach – start with a storage pool and place objects in it
  - Object just a container of data
  - No way to navigate the pool to find objects (no directory structures, few services)
  - Computer-oriented, not user-oriented
- Typical sequence
  - Create an object within the pool, receive an object ID
  - Access object via that ID
  - Delete object via that ID
- Object storage management software like *Hadoop file system* (**HDFS**) and **Ceph** determine where to store objects, manages protection
  - Typically by storing N copies, across N systems, in the object storage cluster
  - Horizontally scalable
  - Content addressable, unstructured

Summary

- Hard disk drives and nonvolatile memory devices are themajor secondary storage I/O units on most computers. Modern secondary storage is structured as large one-dimensional arrays of logical blocks.
- Drives of either type may be attached to a computer system in one of three ways: (1) through the local I/O ports on the host computer, (2) directly connected to motherboards, or (3) through a communications network or storage network connection.
- Requests for secondary storage I/O are generated by the file system and by the virtual memory system. Each request specifies the address on the device to be referenced in the form of a logical block number.

Summary (Cont.)

- Disk-scheduling algorithms can improve the effective bandwidth of HDDs, the average response time, and the variance in response time. Algorithms such as SCAN and C-SCAN are designed to make such improvements through strategies for disk-queue ordering. Performance of disks cheduling algorithms can vary greatly on hard disks. In contrast, because solid-state disks have no moving parts, performance varies little among scheduling algorithms, and quite often a simple FCFS strategy is used.
- Data storage and transmission are complex and frequently result in errors. Error detection attempts to spot such problems to alert the system for corrective action and to avoid error propagation. Error correction can detect and repair problems, depending on the amount of correction data available and the amount of data that was corrupted.

Summary (Cont.)

- Storage devices are partitioned into one or more chunks of space. Each partition can hold a volume or be part of a multidevice volume. File systems are created in volumes.
- The operating system manages the storage device's blocks. New devices typically come pre-formatted. The device is partitioned, file systems are created, and boot blocks are allocated to store the system's bootstrap program if the device will contain an operating system. Finally, when a block or page is corrupted, the system must have a way to lock out that block or to replace it logically with a spare.
- An efficient swap space is a key to good performance in some systems. Some systems dedicate a raw partition to swap space, and others use a file within the file system instead. Still other systems allow the user or system administrator to make the decision by providing both options.

Summary (Cont.)

- Because of the amount of storage required on large systems, and because storage devices fail in various ways, secondary storage devices are frequently made redundant via RAID algorithms. These algorithms allow more than one drive to be used for a given operation and allow continued operation and even automatic recovery in the face of a drive failure. RAID algorithms are organized into different levels; each level provides some combination of reliability and high transfer rates.
- Object storage is used for big data problems such as indexing the Internet and cloud photo storage. Objects are self-defining collections of data, addressed by object ID rather than file name. Typically it uses replication for data protection, computes based on the data on systems where a copy of the data exists, and is horizontally scalable for vast capacity and easy expansion.

End of Chapter 10.C.