Mass -Storage Structure

Dr Tran Duc Minh and Dr Hung Tran

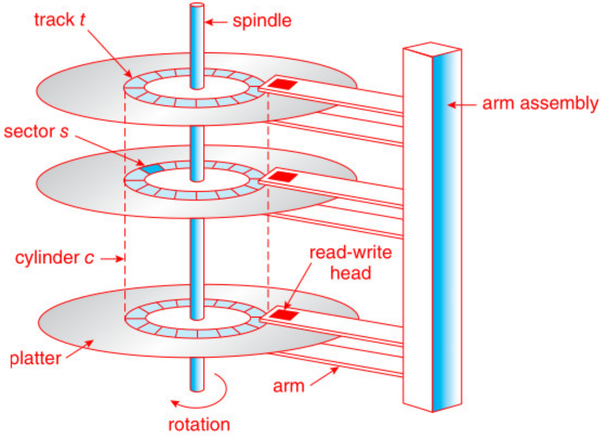
DATCOM Lab
Faculty of Data Science and Artificial Intelligence
College of Technology, National Economics University
Email: minhdt@neu.edu.vn, hung.tran@neu.edu.vn

May 5, 2025

Outline

Overview

- Bulk of secondary storage for modern computers is **hard disk drives (HDDs)** and **nonvolatile memory (NVM) devices**.
- Hard Disk Drives



Hard Disk Drives

- **HDDs** spin platters of magnetically-coated material under moving read-write heads
  - Drives rotate 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**)
  - **Head crash** results from disk head making contact with the disk surface.
- Disks can be removable.

Hard Disk Drives

- Platters range from 0.85" to 14" (historically)
  - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
  - Transfer Rate (theoretical): 6 Gb/sec
  - Effective Transfer Rate (real): 1Gb/sec
  - Seek time from 3ms to 12ms. 9ms common for desktop drives.
  - Average seek time measured or calculated based on 1/3 of tracks.
  - Latency based on spindle speed
    - ★ $1/(RPM/60) = 60/RPM$
  - Average latency = 1/2 latency

Hard Disk Performance

- **Access Latency** = **Average access time** = average seek time + average 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 .1ms controller overhead =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + 0.031ms = 9.301ms

Nonvolatile Memory Devices

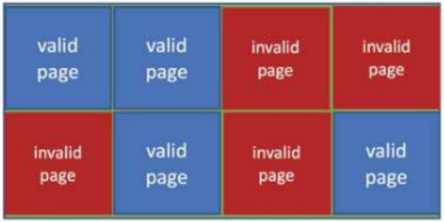
- If disk-drive like, then called solid-state disks (SSDs)
- Other forms include USB drives (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
- Can be more reliable than HDDs
- More expensive per MB
- May have shorter life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

Nonvolatile Memory Devices

- 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 times
  - Life span measured in **drive writes per day (DWPD)**
    - ★ 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** 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



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
- Computers have buffering, caching via RAM, so why RAM drives?
  - Caches / buffers allocated / managed by programmer, operating system, hardware
  - RAM drives under user control
  - Found in all major operating systems
    - Linux `/dev/ram`, macOS `diskutil` to create them, Linux `/tmp` of file system type `tmpfs`
- Used as high speed temporary storage
  - Programs could share bulk date, quickly, by reading/writing to RAM drive

Secondary Storage Connection Methods

- 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)**.
- Most common is SATA
- 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**, **HBAs**)
  - 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

Overview

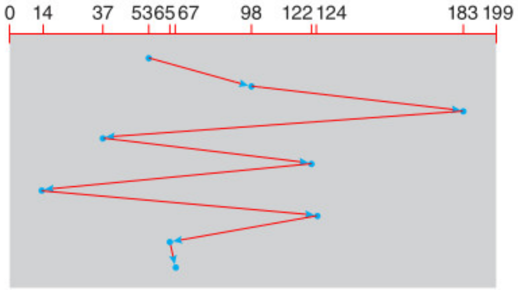
- The operating system is responsible for using hardware efficiently
  - For the disk drives, this means having a fast access time and disk bandwidth.
- Minimize seek time.
- Seek time $\approx$ seek distance
- **Disk bandwidth** is the total number of bytes transferred, divided by the total time between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

Overview

- There are many sources of disk I/O request
    - Operating System
    - 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 LBAs, handle sorting of requests
        - ★ Some of the algorithms they use described next

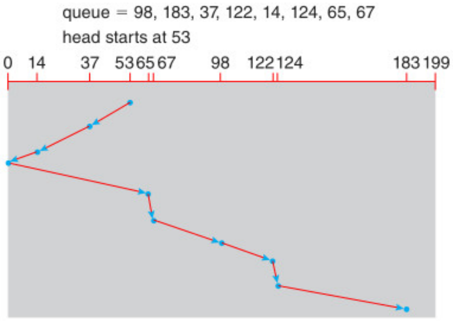
FCFS Scheduling

- The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm (or FIFO).
  - This algorithm is intrinsically fair, but it generally does not provide the fastest service.
- For example: A disk queue with requests for I/O to blocks on cylinders:
  - 98, 183, 37, 122, 14, 124, 65, 67,
  - Head starts at 53
  - Illustration shows total head movement of 640 cylinders



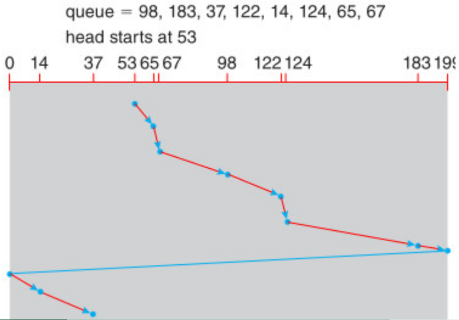
SCAN Scheduling

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues.
- The head continuously scans back and forth across the disk.



C-SCAN Scheduling

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
- Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip.



SSTF Scheduling

- **Shortest Seek Time First (SSTF)** operates by selecting the disk access request with the shortest distance (measured in cylinders) from the current position of the read/write head (disk head) to process first.
- The primary goal is to minimize seek time, which is the time required to move the read/write head from its current position to the cylinder containing the desired data.
- This is a greedy algorithm, prioritizing the nearest request at each step, regardless of the order in which requests arrive.
- For example: Suppose the read/write head is currently at cylinder 50, with access requests at: [180, 20, 90, 110, 70, 150].
  - Possible order: $50 \rightarrow 70 \rightarrow 90 \rightarrow 110 \rightarrow 150 \rightarrow 180$.

Selection of a Disk-Scheduling Algorithm

- SSTF 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 $\times$ read and 2 $\times$ write
    - 1 read and 1 write queue sorted in LBA order, essentially 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 also NOOP and completely fair queueing scheduler (CFQ) 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
    - ★ The throughput of NVM and HDD can be similar in some cases, especially if the workload doesn't fully leverage NVM's strengths (like random I/O).
  - 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
    - ★ Write amplification is an issue with NVM, particularly SSDs. A single write can trigger additional operations like garbage collection, leading to multiple read/write actions, which can reduce NVM's performance advantage over HDDs.

Error Detection and Correction

- Fundamental aspect of many parts of computing (memory, networking, storage)
  - In a computer's RAM, a bit might flip due to cosmic radiation, changing a value from 0 to 1. If undetected, this could cause a program to crash or produce incorrect results.
- Error detection - determines if there a problem has occurred (for example a bit flipping)
  - If detected, can halt the operation
    - ⋆ Once an error is detected, the system can stop the operation to prevent further issues, such as processing corrupted data.
  - Detection frequently done via parity bit
    - ⋆ Each byte in a memory system has a parity bit associated with it that records whether the number of bits in the byte set to 1 is even (parity $= 0$) or odd (parity $= 1$).
    - ⋆ Note that parity is easily calculated by performing an XOR (for "eXclusive OR") of the bits.

Error Detection and Correction

- **Example**: Suppose a memory system uses parity with even parity (parity = 0 when the total number of 1s is even). A data byte is 10110010.
  - **Data bits**: 10110010. The total number of $1s$ is 4 (even). The system uses even parity, the parity bit is set to 0.
    - Calculating parity bit with XOR: XOR of the data bits: $1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$ (result is 0, matching the parity).
  - **Single-bit error case**: Suppose the third bit flips, changing the byte to 10010010. The total number of $1s$ in the 8 data bits is now 3 (odd). XOR: $1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 1$, which does not match the parity bit (0), so the error is detected.
  - **Double-bit error case**: Suppose the third and fourth bits flip, changing the byte to 10000010. The total number of $1s$ in the 8 data bits is again 4 (even). XOR: $1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$, which matches the parity bit (0), so the double-bit error goes undetected.

Error Detection and Correction

- 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

Questions