CISC 3320

Mass Storage: Overview and HDDs

Hui Chen

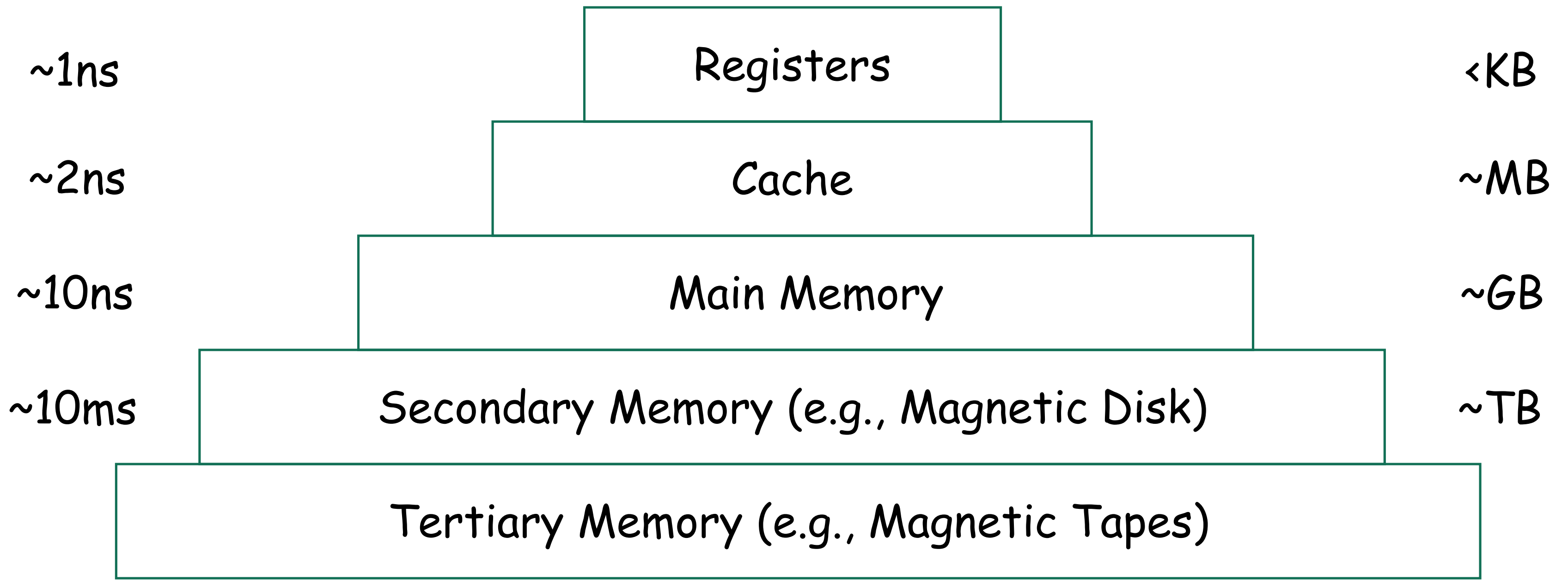
Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Memory Hierarchy


~1ns
Registers
<KB
~2ns
Cache
~MB
~10ns
Main Memory
~GB
~10ms
Secondary Memory (e.g., Magnetic Disk)
~TB
Tertiary Memory (e.g., Magnetic Tapes)

Outline

- Overview of Mass Storage Structure
- HDD Scheduling

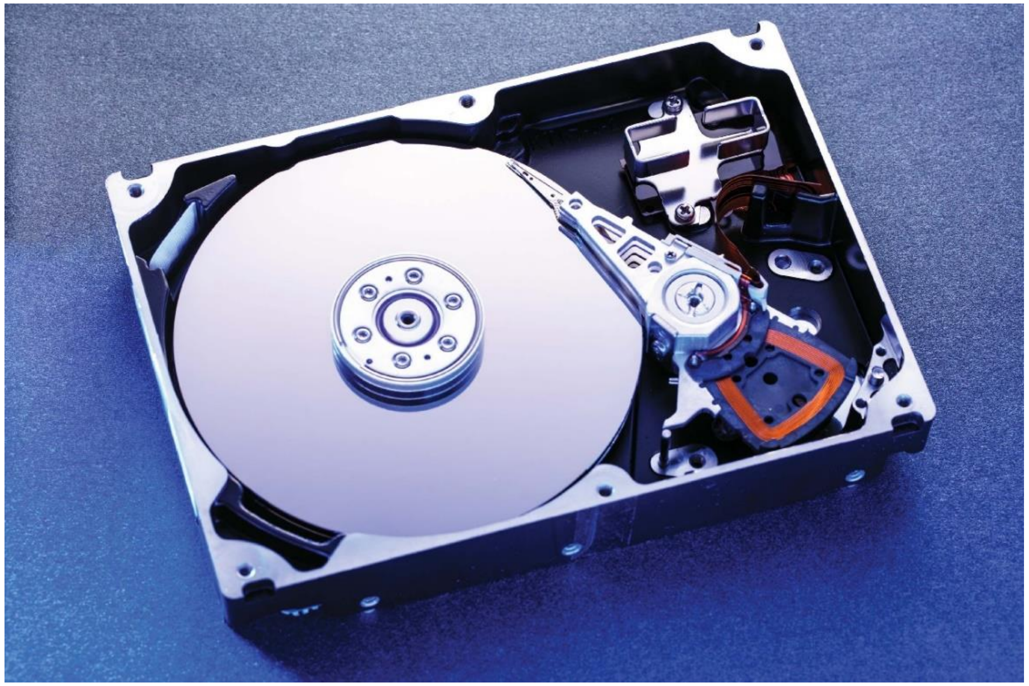
- NVM Scheduling
- Error Detection and Correction
- Storage Device Management
- Swap-Space Management
- Storage Attachment
- RAID Structure

Overview of Mass Storage Devices

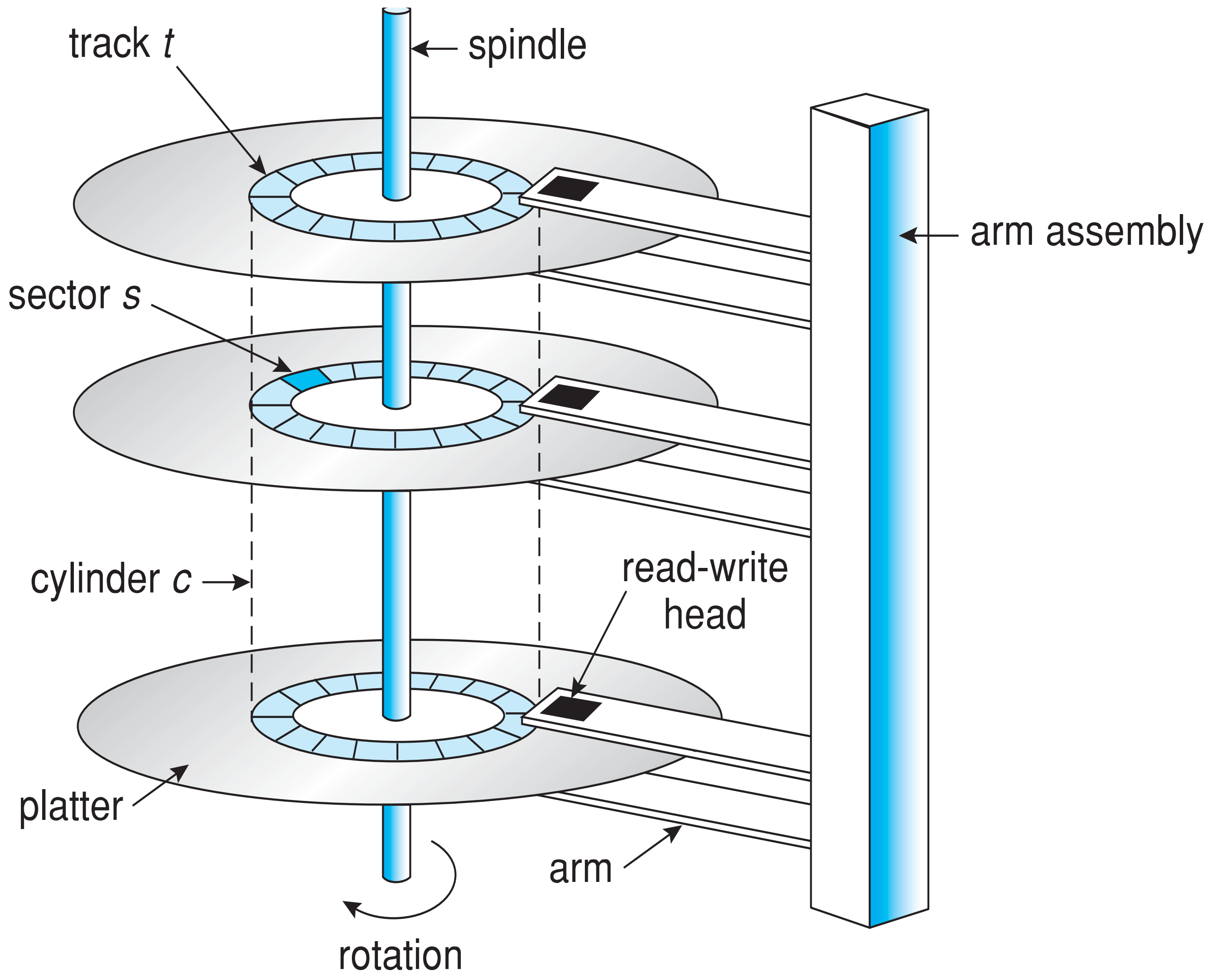
- Bulk of secondary storage for modern computers are
  - Hard disk drives (HDDs)
  - Nonvolatile memory (NVM) devices
- Frequently used as a mass storage device
  - Volatile memory
- Once used as a secondary storage
  - Magnetic tapes

Hard Disk Drives

- HDDs spin platters of magnetically-coated material under moving read-write heads



Moving-head Disk Mechanism



HDD Performance

- Rotational speed
- Transfer rate
- Positioning time (or random access time)

Rotation Speed

- A disk drive motor spins it at high speed.
- Most drives rotate 60 to 250 times per second
- Common drives spin at 5,400, 7,200, 10,000, and 15,000 Rotations Per Minute (RPM)
- Some drives power down when not in use and spin up upon receiving an I/O request.

Transfer rate

- Transfer rate is rate at which data flow between drive and computer
    - Commonly ~ gigabits / second (Gb/s)

Random Access Time

- Positioning time (or random access time) consists of two parts
  - Seek time
    - time to move disk arm to desired cylinder
  - Rotational latency
    - time for desired sector to rotate under the disk head (rotational latency)

HDD Characteristics: Size & Capacity: Examples

- Platters range from .85” to 14” (historically)
    - Commonly 3.5”, 2.5”, and 1.8”
- Range from ~100GB to ~10TB per drive

HDD Performance: Transfer Rate: Examples

- Transfer Rate
    - Theoretical/stated, 6 Gb/sec
- Effective Transfer Rate
    - Real/can be achieved/actual, ~1Gb/sec

HDD Performance: Random Access Time: Examples

- Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated for movement between tracks
- Rotational latency based on spindle speed
  - 1 / (RPM / 60) = 60 / RPM
  - Average rotational latency = ½ rotational latency

HDD Performance: Access Latency: Examples

- **Access latency** = **Average access time** = average seek time + average (rotational) latency
  - Examples
    - For fastest disk 3ms + 2ms = 5ms
    - For slow disk 9ms + 5.56ms = 14.56ms

Average I/O Time: Example

- Transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead
    - Average access time = average seek time + average latency = 5 + ½ 1/(7200/60) 1000 ≈ 5 ms + 4.17 ms
    - Transfer time = 4KB / 1Gb/s ≈ 0.031 ms
    - Average I/O time for 4KB block ≈ (5ms + 4.17ms) + 0.1ms + 0.031 ms = 9.301 ms
- Which part is dominant?

Issues with HDD

- Seek time is dominant
- How to minimize seek time?
  - Disk scheduling/Disk arm scheduling algorithms

Questions?

- Hard disk drives
- Characteristics and performance?

Using HDD Efficiently

- The operating system is responsible for using hardware efficiently
- For the disk drives, this means having a fast access time and disk bandwidth
- Access time = seek time + rotational latency + data transfer time + controller overhead
- Disk **bandwidth**
  - (the total number of bytes transferred)/(the total time between the first request for service and the completion of the last transfer)

Seek Time

- Recall the factors of a disk block read/write performance
  - Seek time (the time to move the arm to the proper cylinder)
  - Rotational latency (how long for the proper sector to come under the head)
  - Data transfer time
  - Controller overhead
- The seek time often dominates.
- Can we reduce the seek time?
- Seek time ≈ seek distance, can we reduce seek distance?
- When seek time becomes less, data bandwidth becomes greater

Disk I/O Request Queue and Disk Scheduling

- There are many sources of disk I/O request
  - OS, system processes, and 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
- Optimization is to reduce seek distance → Disk scheduling/disk arm scheduling algorithms

Disk Scheduling Problem

- Assumption
  - A disk driver accepts bursts of access requests
- Problem
  - In which order should these requests be carried out to minimize the seek time?
    - Average seek time?
    - Overall seek time?
- Algorithms
  - FCFS
  - SCAN
  - C-SCAN
  - And more

Example Requests for Analysis

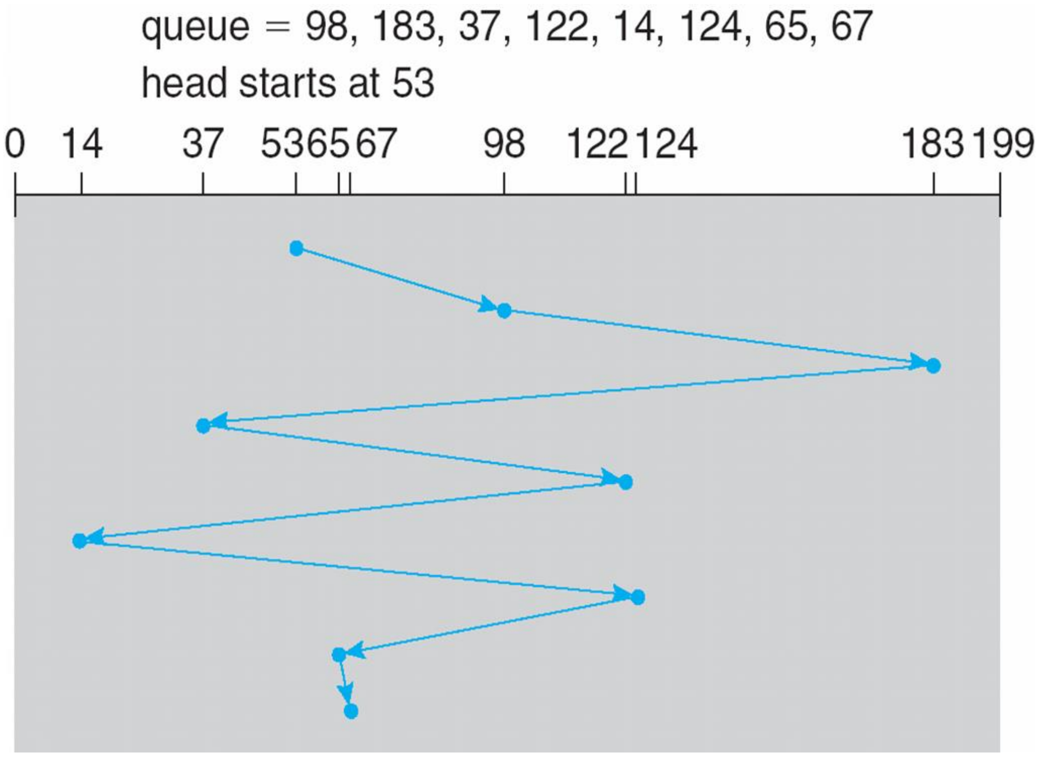
- A request queue (0-199): a list of HDD cylinder numbers

  98, 183, 37, 122, 14, 124, 65, 67

- Head pointer 53 at the beginning

FCFS

- Illustration shows total head movement of 640 cylinders

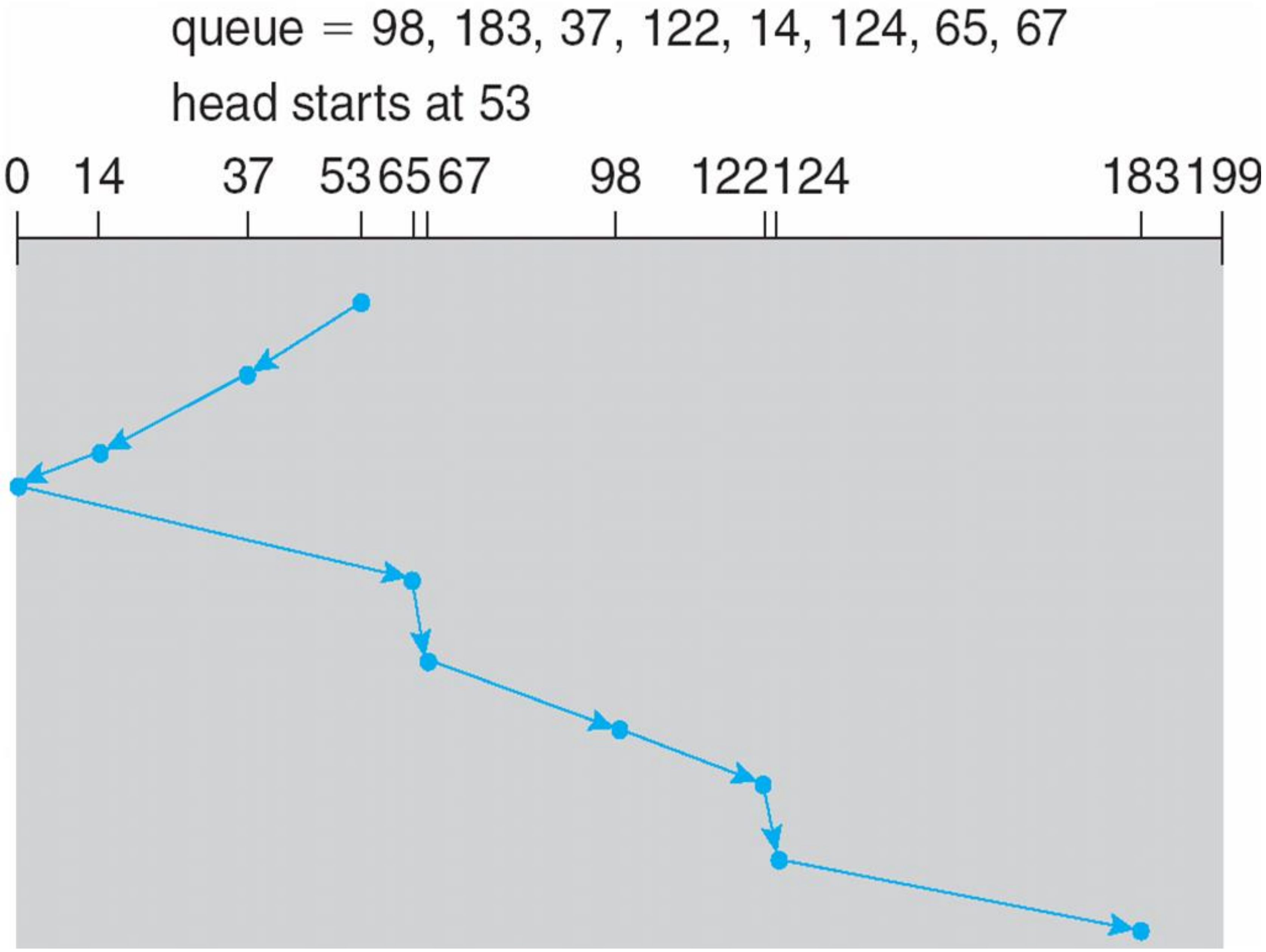


SCAN

- **SCAN algorithm** Sometimes called the **elevator 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..

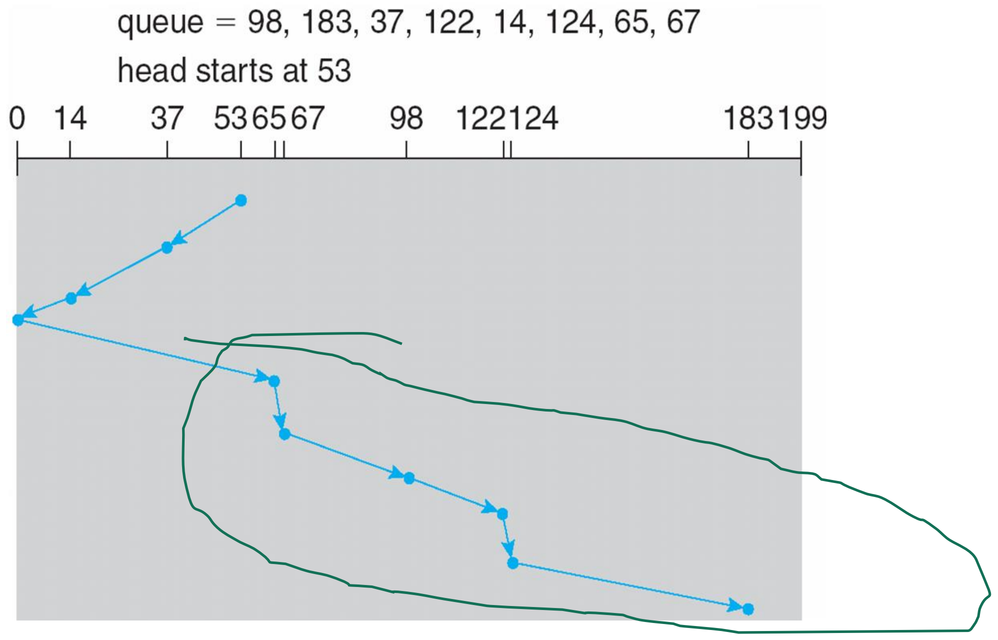
SCAN: Example

- Illustration shows total head movement of 208 cylinders


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN: Observation

- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

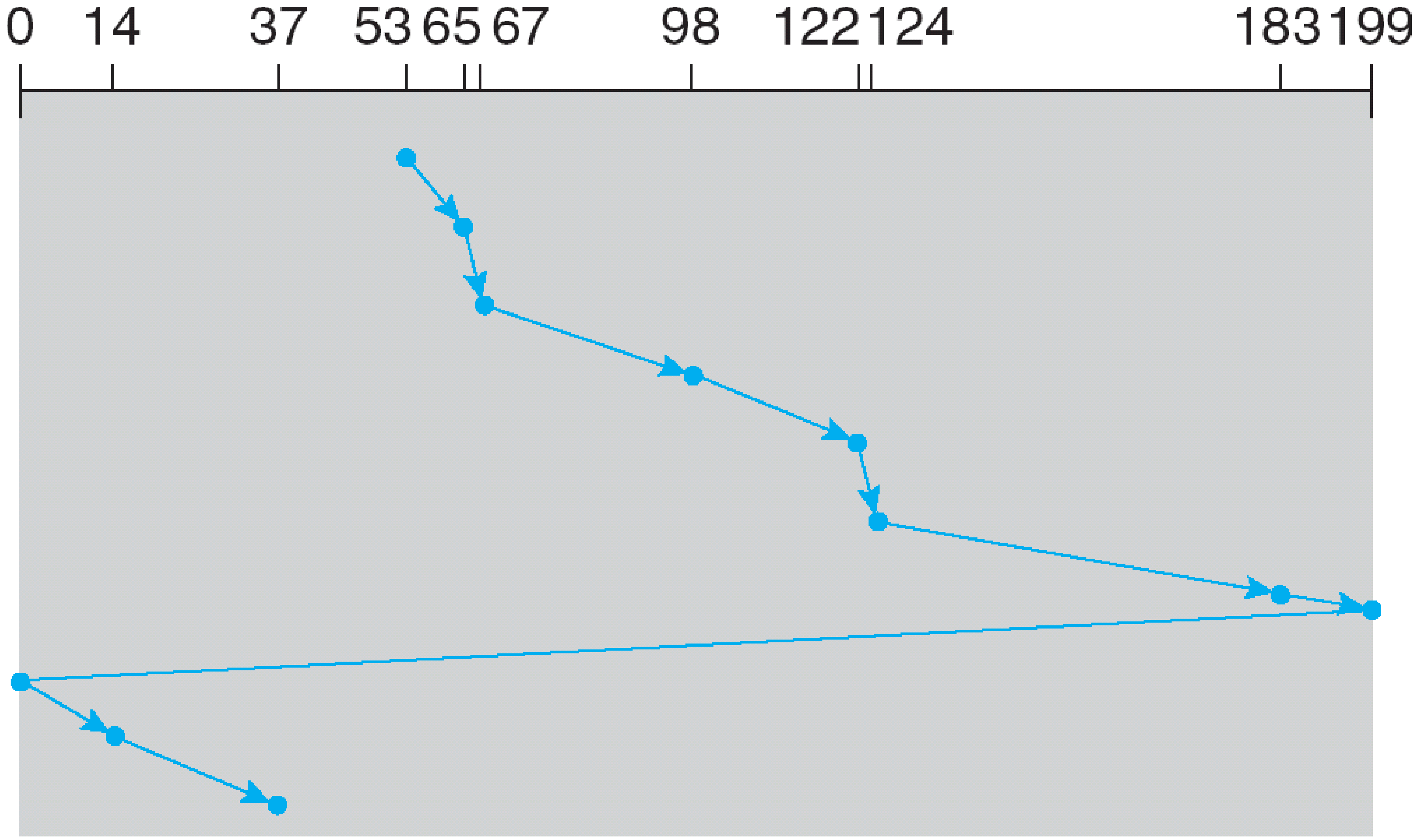


Circular-Scan (C-SCAN)

- 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
- Provides a more uniform wait time than SCAN

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



Selecting a Disk-Scheduling Algorithm

- 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, 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
  - NOOP for CPU-bound systems using fast storage such as NVM devices,
  - CFA for SATA HDDs

Questions?

- Disk scheduling/Disk arm scheduling
- FCFS
- SCAN
- C-SCAN
- And more
- Which one to use?