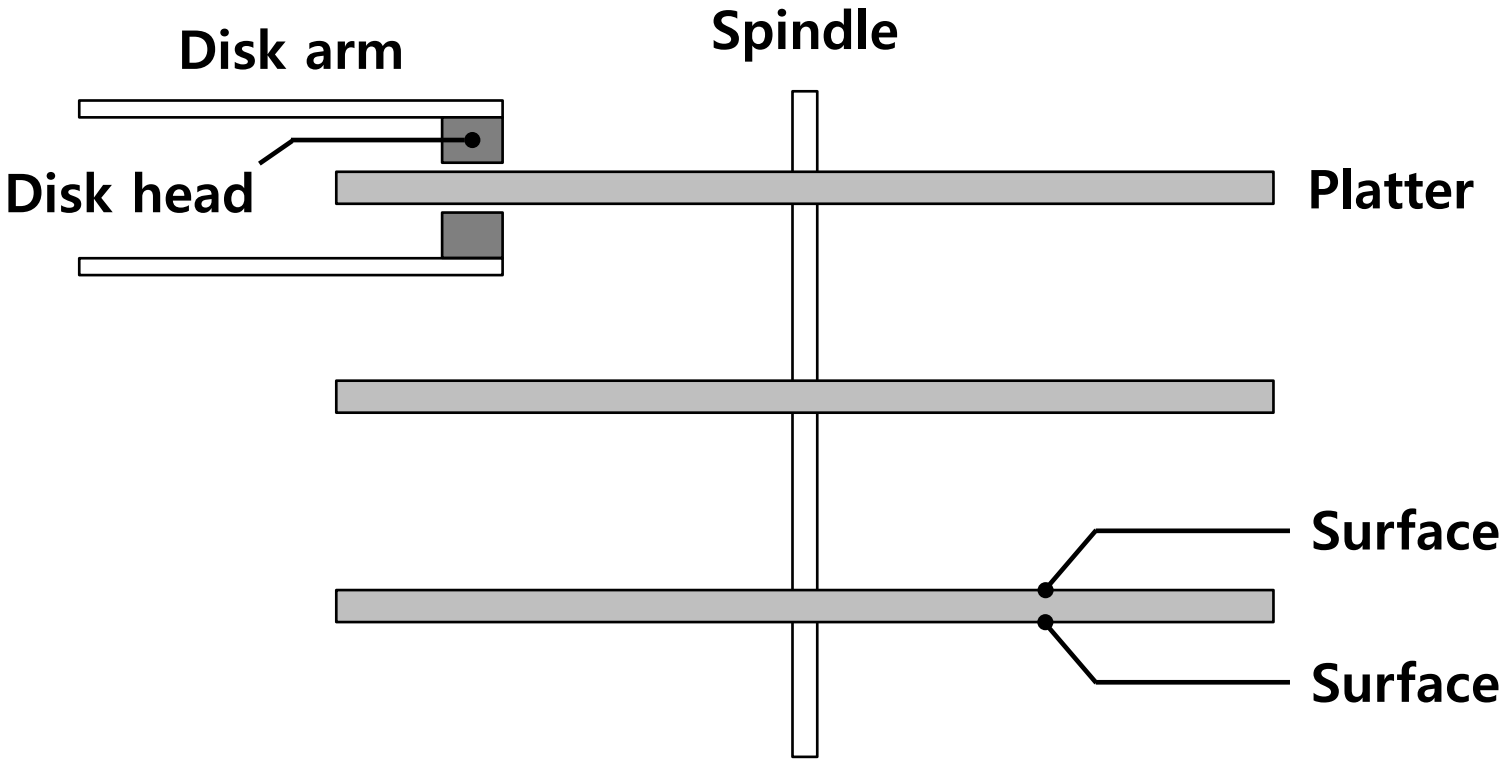
# Operating Systems

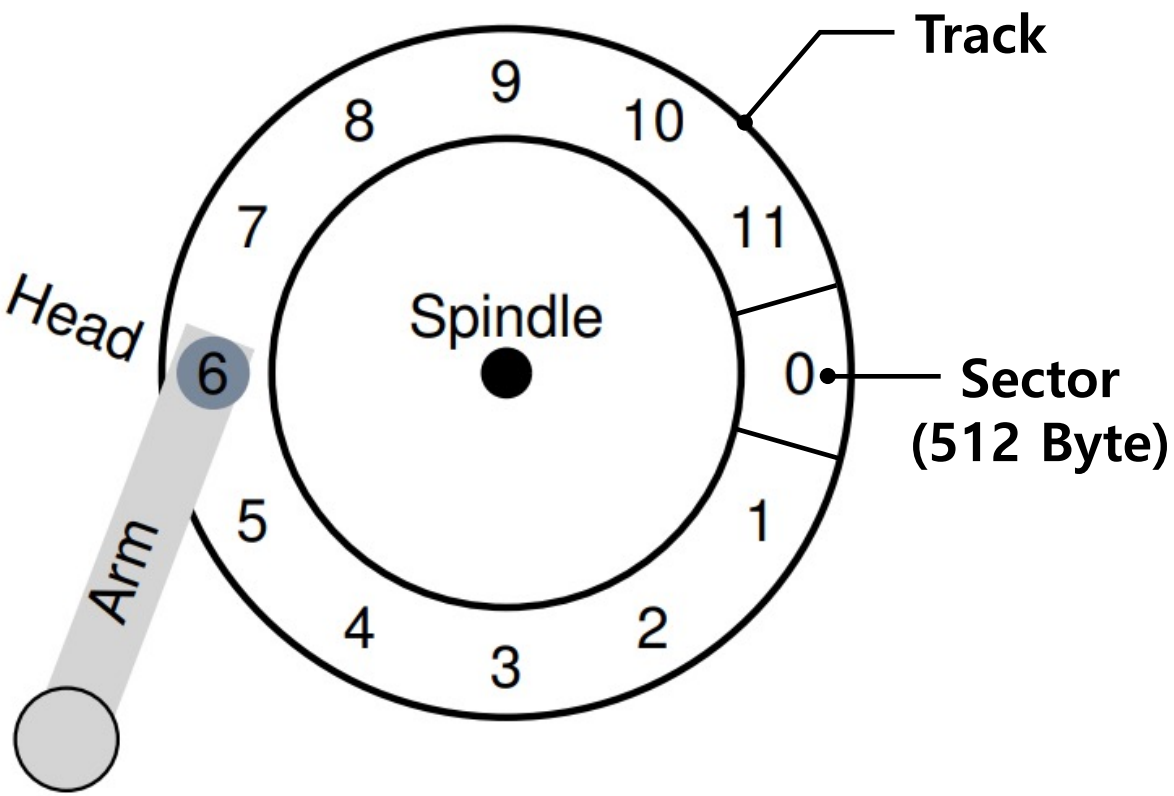
Lecture 12

# 37. Hard Disk Drives

# Base Geometry



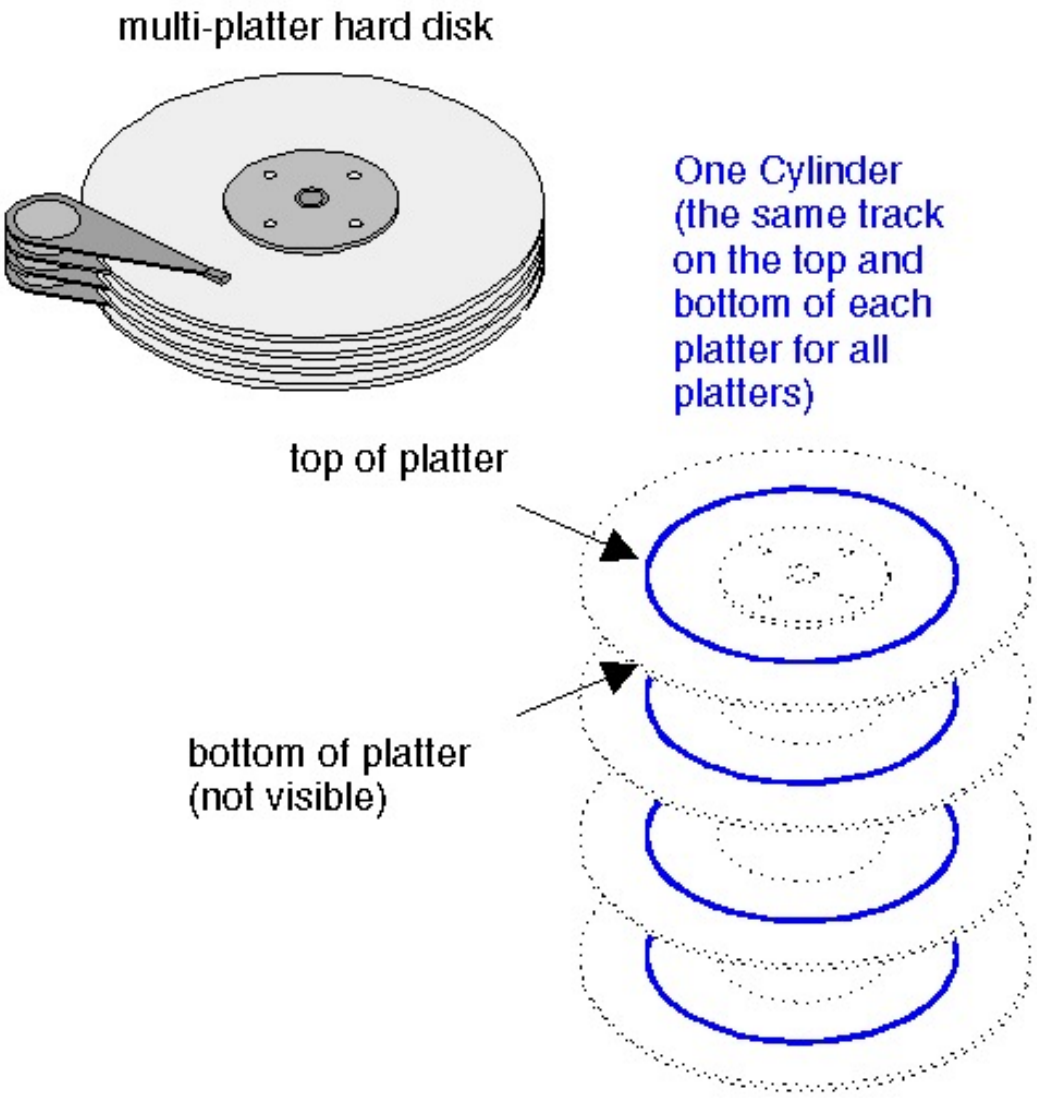
# Base Geometry (Cont.)




HDD demo

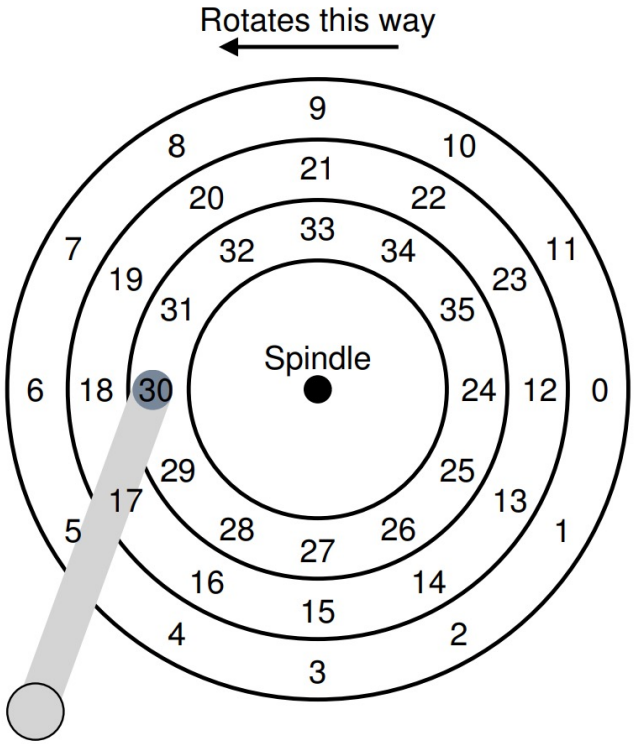
- https://www.youtube.com/watch?v=L0nbo1VOF4M
- https://www.youtube.com/watch?v=9eMWG3fwiEU&feature=youtu.be&t=30s

# Base Geometry (Cont.)



# A Simple Disk Drive

- Rotation Delay

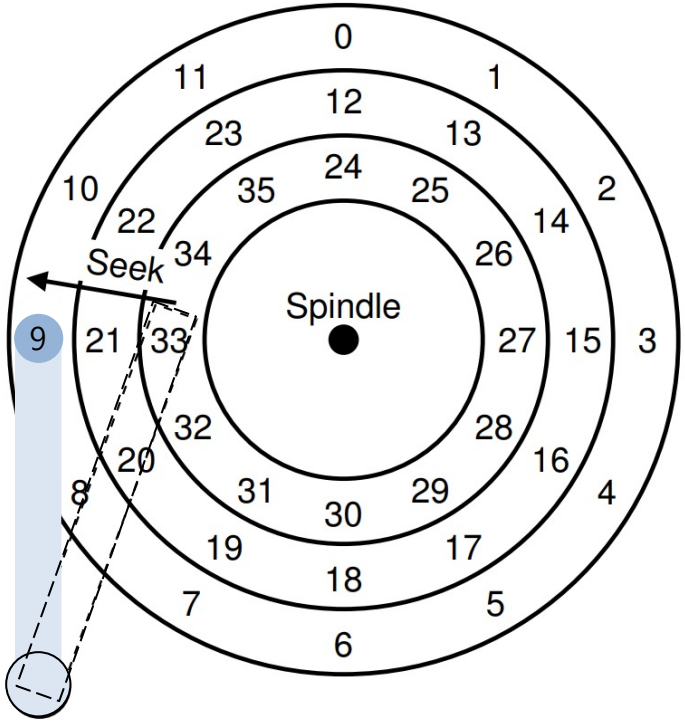



If the full rotation delay is $R$,

the rotation delay of (30→24) is $\frac{R}{2}$

# A Simple Disk Drive (Cont.)

- Seek Time

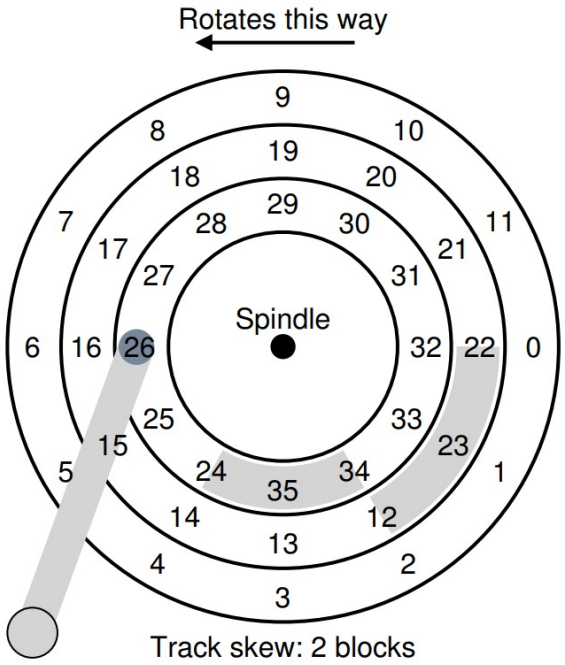



Phases of seek

*Acceleration* → *Coasting* → *Deceleration* → *Settling time*
(about 0.5~2 ms)

# A Simple Disk Drive (Cont.)

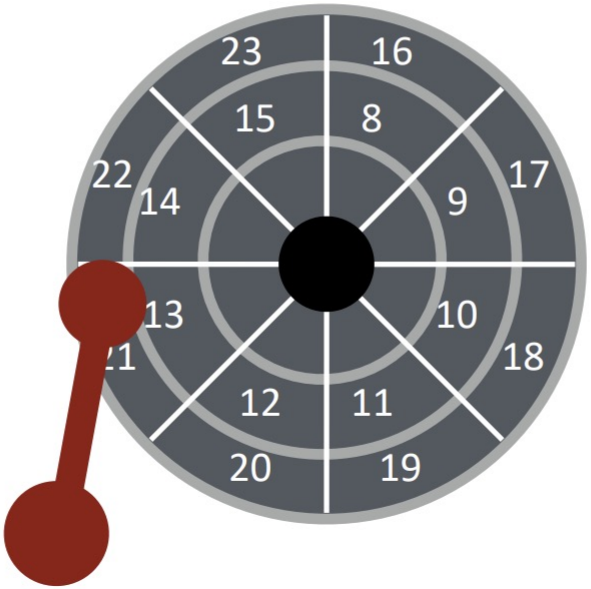
- Track skew
  - Make sure that sequential reads can be properly serviced even when crossing track boundaries
  - Without such skew, the head would be moved to the next track but the desired next block would have already rotated under the head



# A Simple Disk Drive (Cont.)

- Track skew

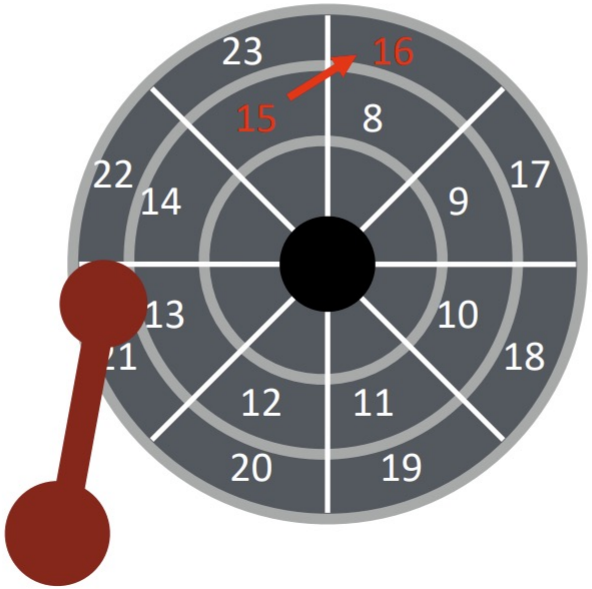
Imagine sequential reading. How should sectors numbers be laid out on disk?



# A Simple Disk Drive (Cont.)

- Track skew

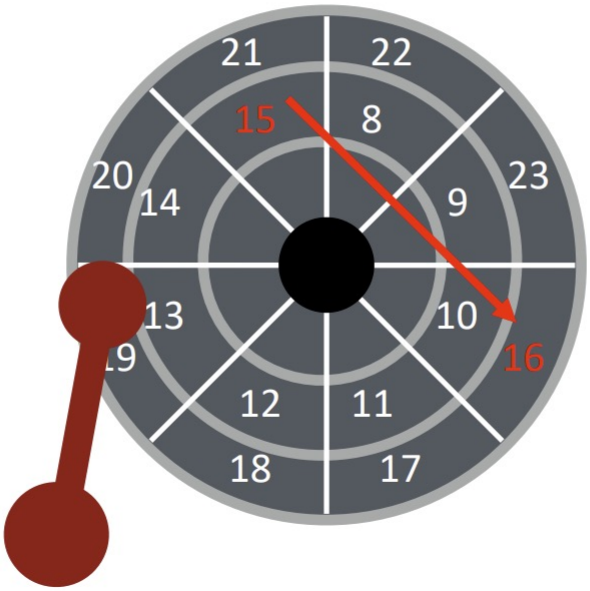
When reading 16 after 15, the head won't settle quick enough, so we need to do a rotation.



# A Simple Disk Drive (Cont.)

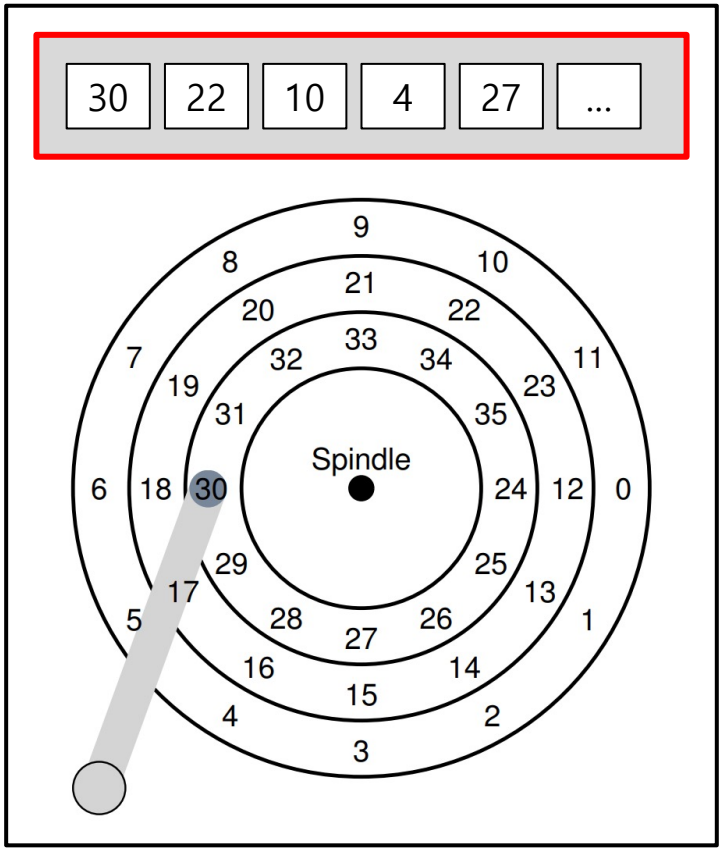
- Track skew

Enough time to settle now!



# A Simple Disk Drive (Cont.)

- Cache (Track buffer)



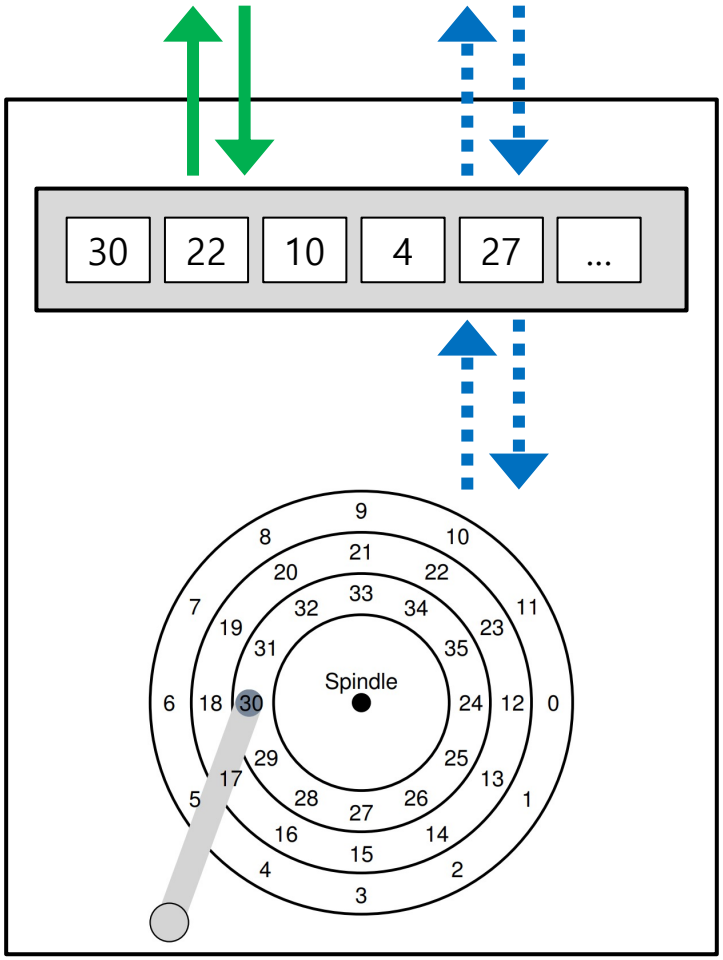

Small amount of memory
(usually around 8 or 16MB)

Hold data read from or written to the disk

Allow the drive to quickly respond to requests

# A Simple Disk Drive (Cont.)

- Cache (Track buffer)




Write-Back

Acknowledge the write has completed when it has put the data in its memory

Write-Through

Acknowledge after the write has actually been written to disk

# I/O Time: Doing The Math

- I/O Time

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

- I/O Rate

$$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$

# I/O Time: Doing The Math (Cont.)

- 4KB Random Read Example

| | Cheetah 15K.5 | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Average Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 16/32 MB |
| Connects via | SCSI | SATA |

$T_{seek}$ = 4ms

$T_{rotation}$ = 15,000 RPM(= 250RPS = 4ms / 1 rotation) / 2
= 2ms

$T_{transfer}$ = 4KB / 125(MB/s)
= 30us

$T_{I/O}$ = 4ms + 2ms + 30us ≒ 6ms

$R_{I/O}$ = 4KB / 6ms = 0.66MB/s

# I/O Time: Doing The Math (Cont.)

- 4KB Random Read Example (Cont.)

| | Cheetah 15K.5 | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Average Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 16/32 MB |
| Connects via | SCSI | SATA |

$T_{seek}$ = 9ms

$T_{rotation}$ = 7,200 RPM(= 120RPS = 8ms / 1 rotation) / 2
= 4ms

$T_{transfer}$ = 4KB / 105(MB/s)
= 38us

$T_{I/O}$ = 9ms + 4ms + 38us ≒ 13ms

$R_{I/O}$ = 4KB / 13ms = 0.31MB/s

# I/O Time: Doing The Math (Cont.)

- 100MB Sequential Read Example

| | Cheetah 15K.5 | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Average Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 16/32 MB |
| Connects via | SCSI | SATA |

$T_{seek}$ = 4ms

$T_{rotation}$ = 15,000 RPM(= 250RPS = 4ms / 1 rotation) / 2
= 2ms

$T_{transfer}$ = 100MB / 125(MB/s)
= 800ms

$T_{I/O}$ = 4ms + 2ms + 800ms = 806ms ≒ 800ms

$R_{I/O}$ = 100MB / 800ms = 125MB/s

# I/O Time: Doing The Math (Cont.)

- 100MB Sequential Read Example (Cont.)

| | Cheetah 15K.5 | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Average Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 16/32 MB |
| Connects via | SCSI | SATA |

$T_{seek}$ = 9ms

$T_{rotation}$ = 7,200 RPM(= 120RPS = 8ms / 1 rotation) / 2
= 4ms

$T_{transfer}$ = 100MB / 105(MB/s)
= 950ms

$T_{I/O}$ = 9ms + 4ms + 950ms = 963ms ≒ 950ms

$R_{I/O}$ = 100MB / 950ms = 105MB/s

# I/O Time: Doing The Math (Cont.)

| | Cheetah | Barracuda |
|---|---|---|
| $R_{I/O}$ Random | 0.66 MB/s | 0.31 MB/s |
| $R_{I/O}$ Sequential | 125 MB/s | 105 MB/s |

Performance vs Capacity

| | Cheetah | Barracuda |
|---|---|---|
| $R_{I/O}$ Random | 0.66 MB/s | 0.31 MB/s |
| $R_{I/O}$ Sequential | 125 MB/s | 105 MB/s |

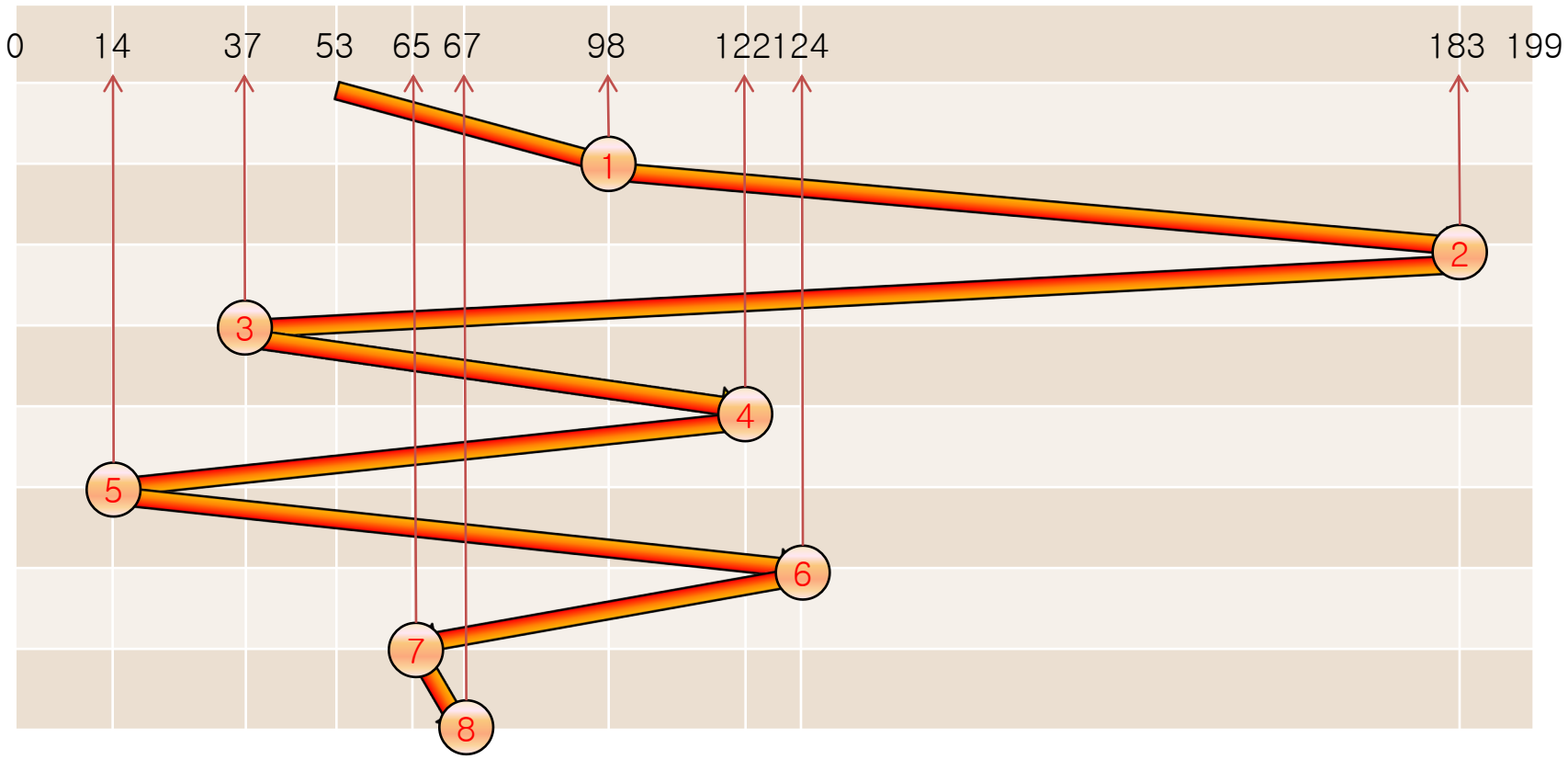
Random Read vs Sequential Read

# Disk Scheduling: FCFS

How to order the services for the requests in the queue?

Illustration shows total head movement of 640 cylinders.

**Queue = 98, 183, 37, 122, 14, 124, 65, 67**
**Head starts at 53**

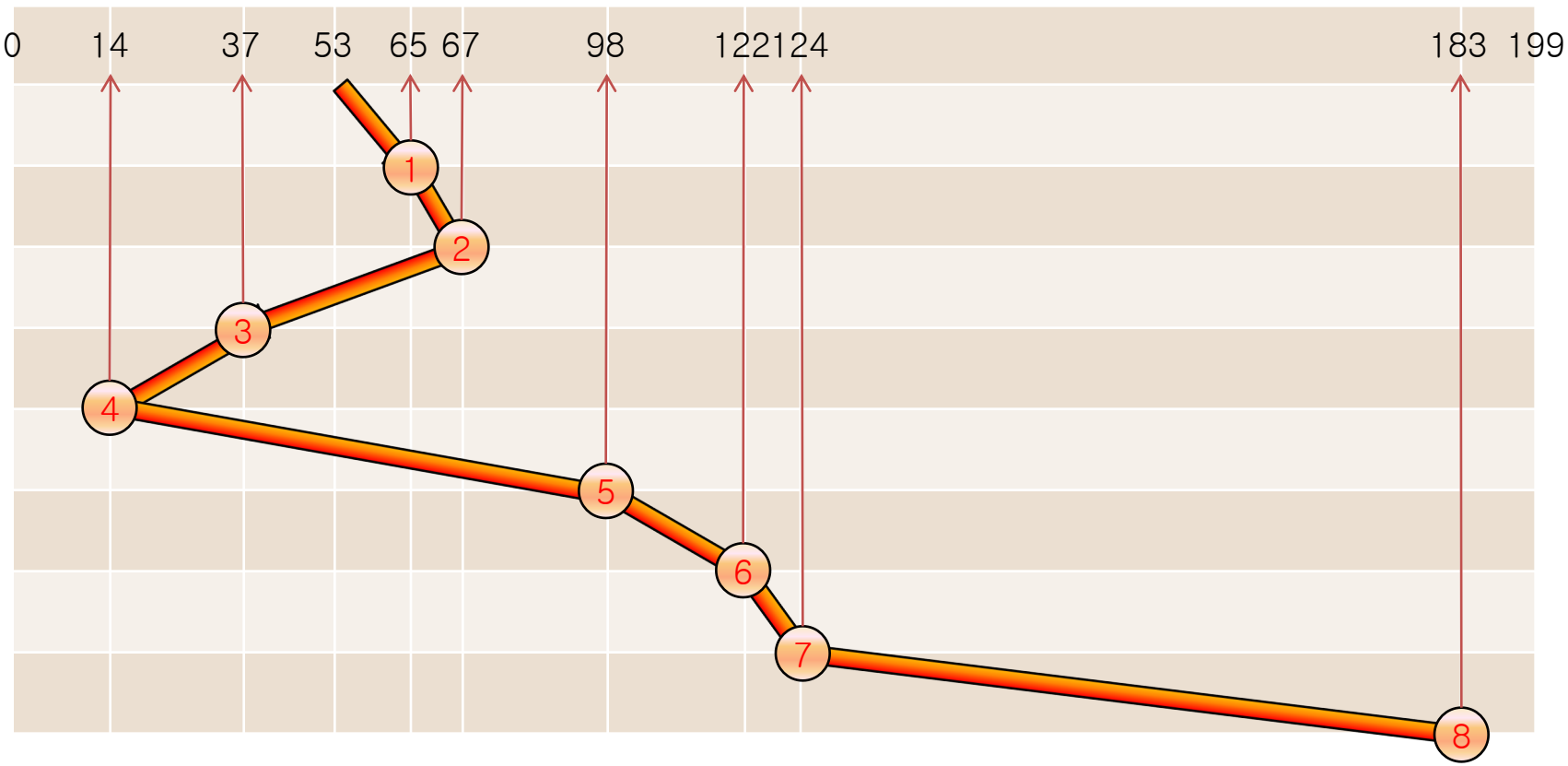


# SSTF (Shortest Seek Time First)

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.

# SSTF (Cont.)

**Queue = 98, 183, 37, 122, 14, 124, 65, 67**
**Head starts at 53**

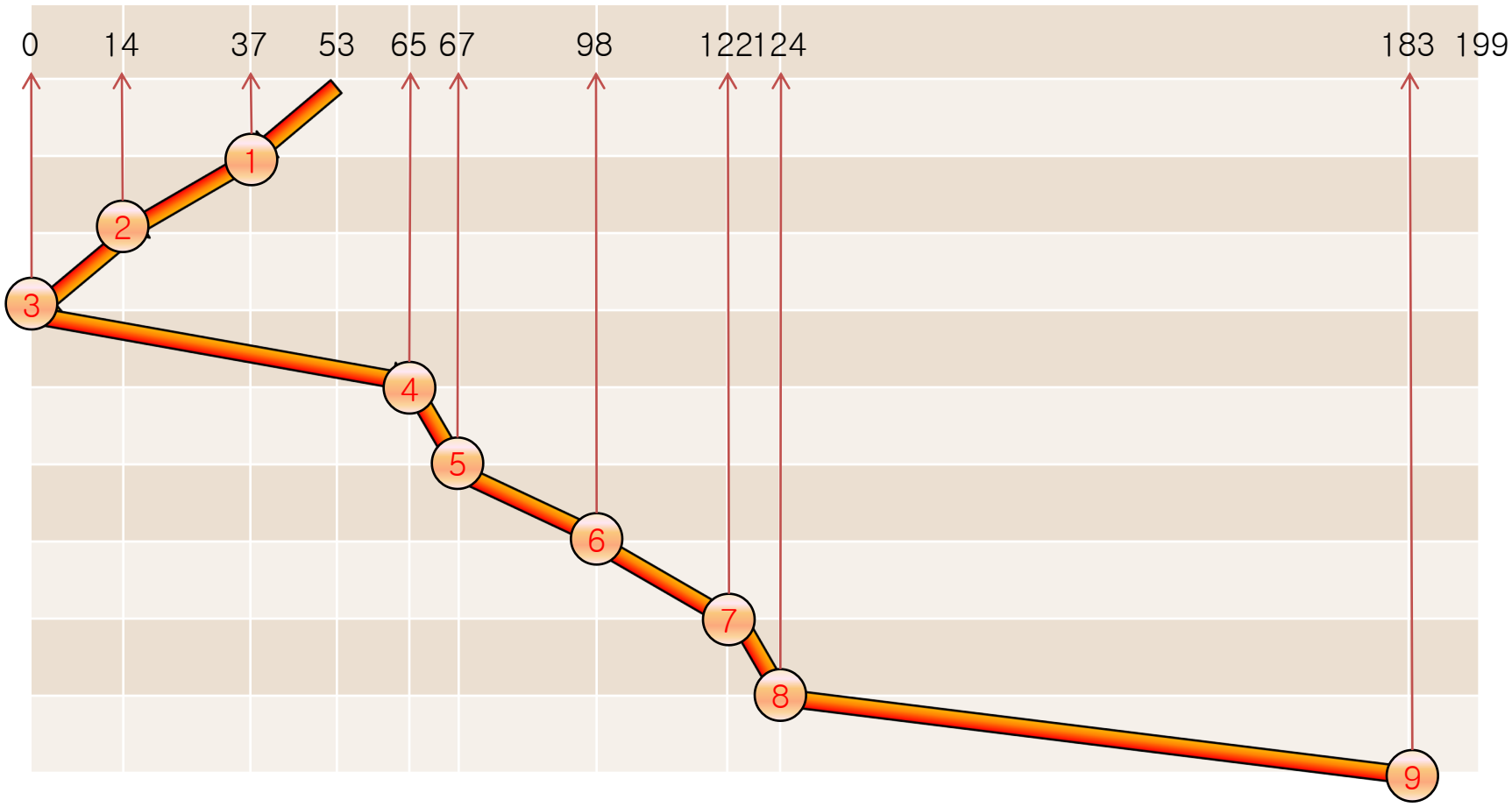


# SCAN

- 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.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

# SCAN (Cont.)

**Queue = 98, 183, 37, 122, 14, 124, 65, 67**
**Head starts at 53**

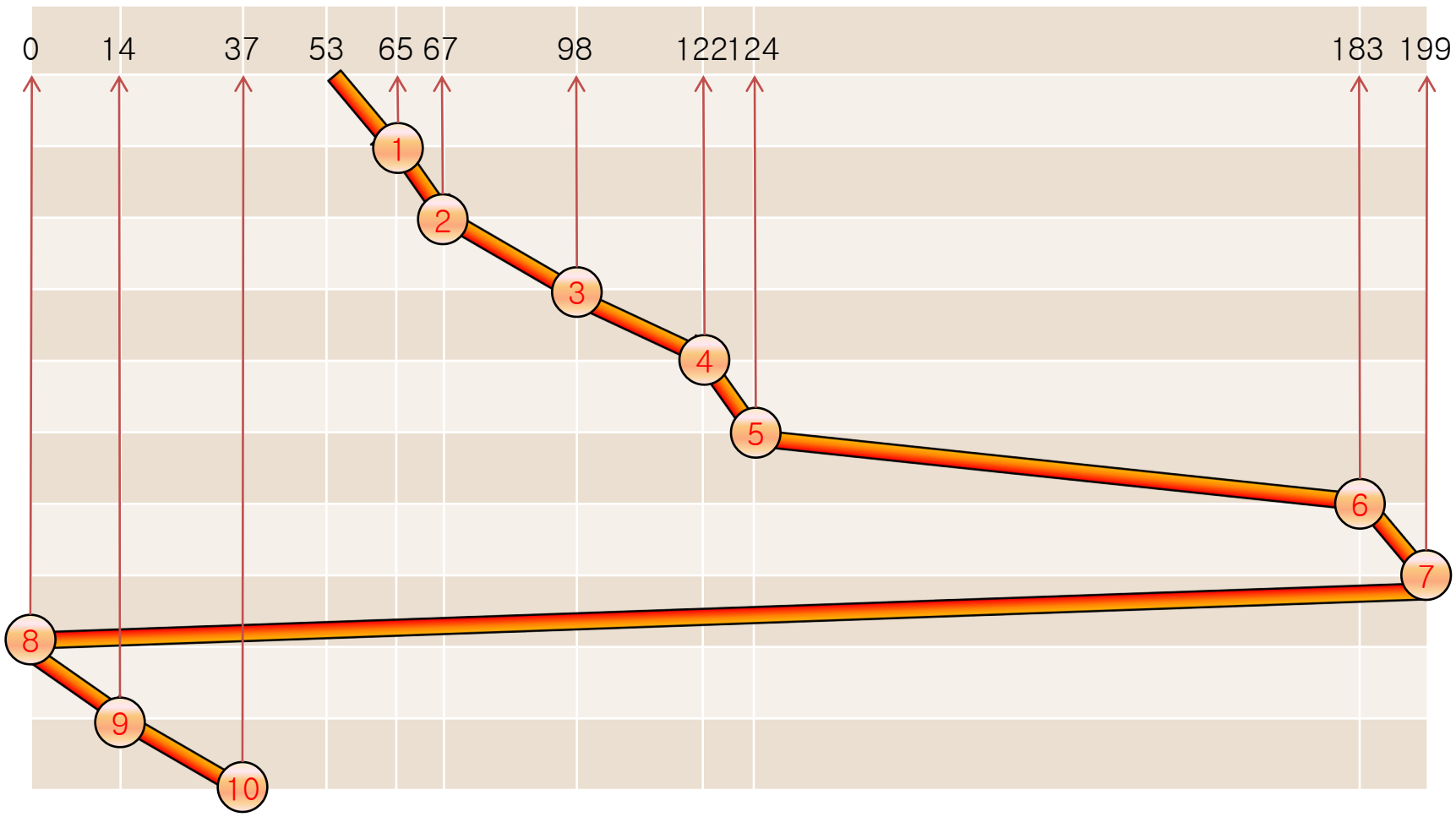


# C-SCAN (Circular-SCAN)

- Only sweeps from outer-to-inner, and then resets at the outer track to begin again.
- 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.
- Provides a more uniform wait time than SCAN.
  - Doing so is a bit more fair to inner and outer tracks, as pure back-and-forth SCAN favors the middle tracks
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

# C-SCAN (Cont.)

**Queue = 98, 183, 37, 122, 14, 124, 65, 67**
**Head starts at 53**

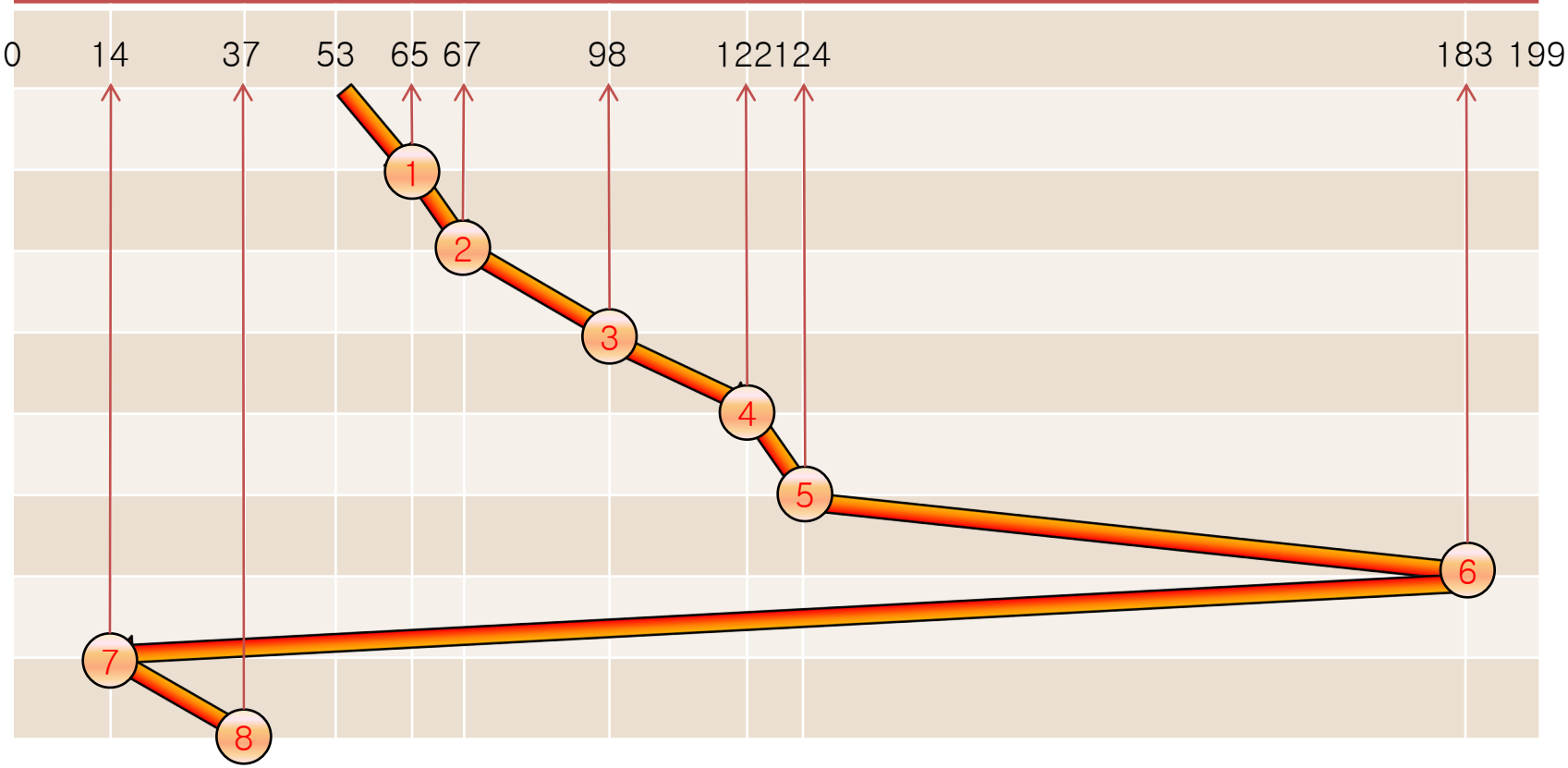


# C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without going all the way to the end of the disk.

# C-LOOK (Cont.)

**Queue = 98, 183, 37, 122, 14, 124, 65, 67**
**Head starts at 53**



# Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN, C-SCAN and C-LOOK perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or C-LOOK is a reasonable choice for the default algorithm.

# 38. RAID

# RAID (Redundant Array of Inexpensive Disks)

- **RAID** is to use multiple disks to build **faster, bigger, and more reliable** disk system.
- RAID is arranged into six different levels.
  - RAID Level 0: Striping multiple disks
  - RAID Level 1: Use mirroring
  - RAID Level 4, level 5: Parity based redundancy

# Evaluation

- Capacity
  - N disks, B blocks per disk
  - N*B blocks in total → How much useful capacity is available to the clients of RAID?
- Reliability
  - How many disk faults can the RAID tolerate
- Performance
  - Read
  - write

# RAID Level 0

- RAID Level 0 is the simplest form as **striping** blocks.
  - Spread the blocks across the disks in a round-robin fashion.

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

**RAID-0: Simple Striping**

# RAID Level 0 (Cont.)

- Chunk size
  - Small chunk:
    - more intra-file parallelism
    - Larger positioning time
  - Large chunk:
    - Reduced intra-file parallelism
    - Smaller positioning time
- An example of RAID Label 0 with a bigger chunk size
  - Chunk size : 2 blocks (8KB)

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | |
|---|---|---|---|---|
| 0 | 2 | 4 | 6 | **chunk size: 2 blocks** |
| 1 | 3 | 5 | 7 | |
| 5 | 10 | 12 | 14 | |
| 9 | 11 | 13 | 15 | |

Striping with a Bigger Chunk Size

# RAID Level 0 Analysis

- Single Disk
  - Average seek time: 7 ms
  - Average rotational delay: 3 ms
  - Transfer rate of disk: 50 MB/s
- Single Disk Performance
  - 10 Mbyte seq. IO, S = $\frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ MB}{(7+3+200)=210\ ms}$ = 47.62 MB /s
  - 10 Kbyte Random IO, R = $\frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ KB}{(7+3+0.195)=10.195\ ms}$ = 0.981 MB /s
- RAID 0
  - Random write, random read = N*R
  - Sequential write, sequential read = N*S

# RAID Level 1

- RAID Level 1 is mirroring
  - Copy more than one of each block in the system.
  - Copy block places on a separate disk to tolerate the disk failures.

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 2 | 2 | 3 | 3 |
| 4 | 4 | 5 | 5 |
| 6 | 6 | 7 | 7 |

**Simple RAID-1: Mirroring**

# RAID level 1

- Capacity N*B/2
- Reliability
  - From one to upto N/2 depending upon the failure disk
- Performance
  - Sequential write: N*S/2 MB/s
  - Sequential read: N*S/2 MB/s
  - Random write: N*R/2 MB/s
  - Random Read: N*R MB/s

# RAID Level 4

- RAID Level 4 is to add redundancy to a disk array as **parity**.

* P: Parity

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | P0 |
| 4 | 5 | 6 | 7 | P1 |
| 8 | 9 | 10 | 11 | P2 |
| 12 | 13 | 14 | 15 | P3 |

**Simple RAID-4 with parity**

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | xor(0,0,1,1)=0 |
| 0 | 1 | 0 | 0 | Xor(0,1,0,0)=1 |

# RAID Level 4 (Cont.)

- The simple RAID Level 4 optimization known as a **Full-stripe write.**
  - Calculate the new value of P0 (Parity 0)
  - Write all of the blocks to the five disks above in parallel
  - Full-stripe writes are the most efficient way

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | P0 |
| 4 | 5 | 6 | 7 | P1 |
| 8 | 9 | 10 | 11 | P2 |
| 12 | 13 | 14 | 15 | P3 |

**Full-stripe Writes In RAID-4**

# Analysis

- Capacity: (N-1)*B
- Sequential read: (N-1)*S
- Sequential write: (N-1)*S for full stripe write

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | P0 |
| 4 | 5 | 6 | 7 | P1 |
| 8 | 9 | 10 | 11 | P2 |
| 12 | 13 | 14 | 15 | P3 |

- Random read: (N-1)*R
- Random write : R/2 (Small write problems!)

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | P0 |
| *4 | 5 | 6 | 7 | +P1 |
| 8 | 9 | 10 | 11 | P2 |
| 12 | *13 | 14 | 15 | +P3 |

# RAID Level 5

- RAID Level 5 is solution of small write problem.
- RAID Level 5's Each stripe is now rotated across the disks.

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | P0 |
| 5 | 6 | 7 | P1 | 4 |
| 10 | 11 | P2 | 8 | 9 |
| 15 | P3 | 12 | 13 | 14 |
| P4 | 16 | 17 | 18 | 19 |

**RAID-5 with Rotated Parity**

# Analysis

- Performance
  - Sequential read, sequential write: (N-1)S
  - Random read: N*R
  - Random write: single write can cause 4 IO's (two read, two write), All N disks can work in parallel: (N*R)/4

# Summary

| | RAID-0 | RAID-1 | RAID-4 | RAID-5 |
|---|---|---|---|---|
| **Capacity** | N | N/2 | N-1 | N-1 |
| **Reliability** | 0 | 1 (for sure)<br>N/2 (if lucky) | 1 | 1 |
| **Throughput** | | | | |
| Sequential Read | NS | (N/2)S | (N-1)S | (N-1)S |
| Sequential Write | NS | (N/2) S | (N-1)S | (N-1)S |
| Random Read | NR | NR | (N-1)R | NR |
| Random Write | NR | (N/2)R | R/2 | (N/4)R |
| **Latency** | | | | |
| Read | D | D | D | D |
| Write | D | D | 2D | 2D |