

Storage Systems

Instructor: Youngjin Kwon

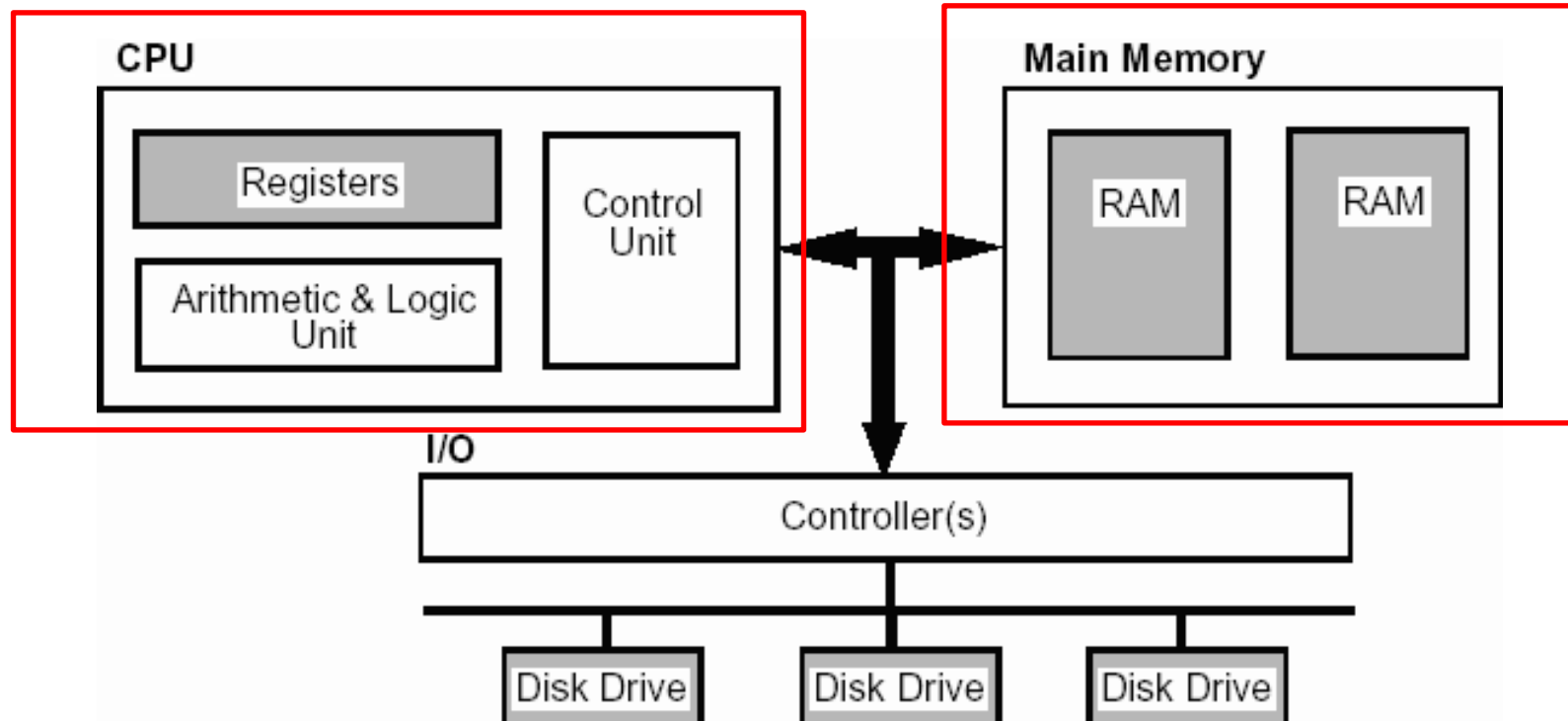
What we've done so far (a resource oriented view)

Abstractions:

Abstractions:

Mechanism and policy for concurrency:

Mechanism and policy:



Main Points

- File systems
 - Useful abstractions on top of physical devices
- Storage hardware characteristics
 - Disks and flash memory
- File system usage patterns

File Systems

- Abstraction on top of persistent storage
 - Magnetic disk
 - Flash memory (e.g., USB thumb drive)
- Devices provide
 - Storage that (usually) survives across machine crashes
 - Block level (random) access
 - Memory: byte level access
 - Large capacity at low cost
 - Relatively slow performance
 - Magnetic disk read takes 10-20M processor instructions

Define a New Abstraction

- Analogy

- Virtual memory is an abstraction of (*Memory*)
- Level of indirection: () \rightarrow ()

- File is an abstraction of (*Storage*)
- Level of indirection: () \rightarrow ()

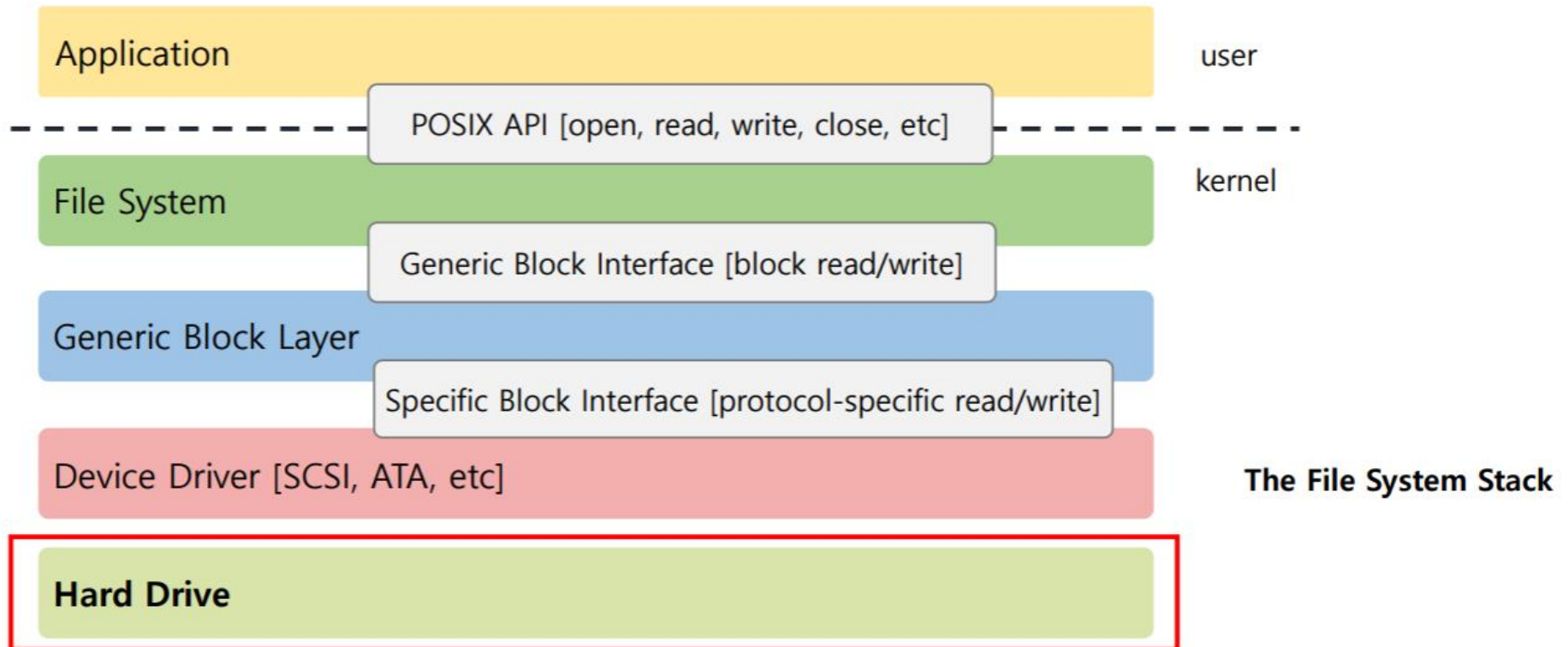
File System Design

- Naming: how to address data block?
- Performance:
- Persistence of data stored in file system:

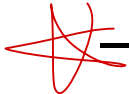
File System Design

- Naming: how to address data block?
 - Mapping file offset to physical block numbers
 - Directories
 - Byte addressable data even though devices are block-oriented
- Performance:
 - Cache data and metadata in storage to DRAM
 - Data placement and data structure organization
- Persistence of data stored in file system:
 - Even if crash happens during an update
 - Even if disk block becomes corrupted
 - Even if flash memory wears out

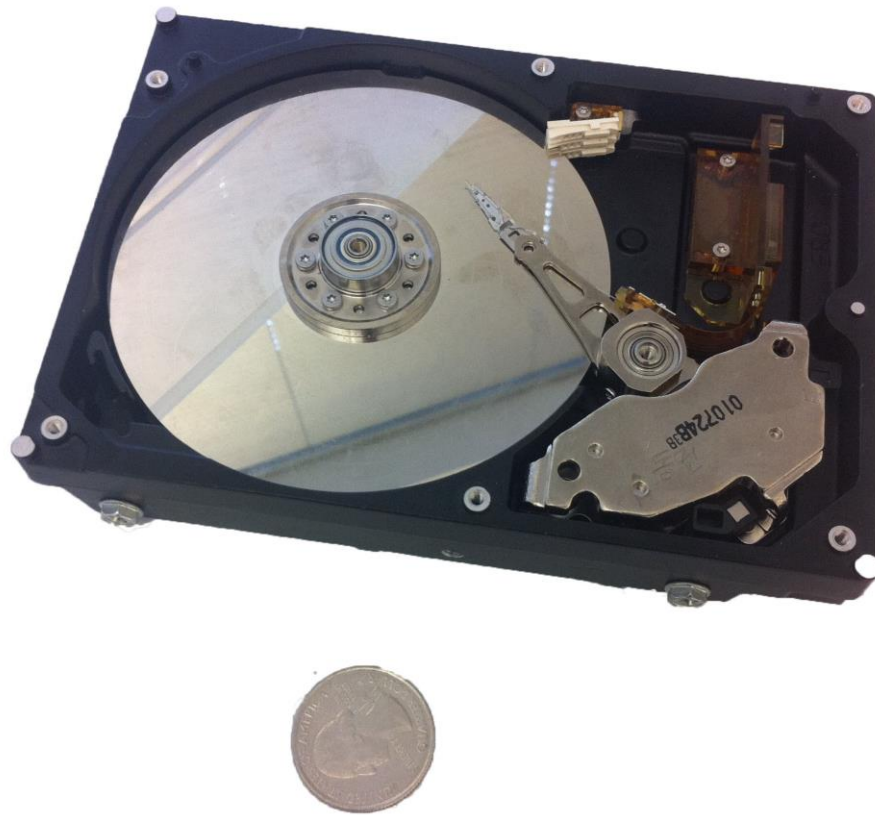
To design a file system,
You first need to know characteristics of
storage devices!

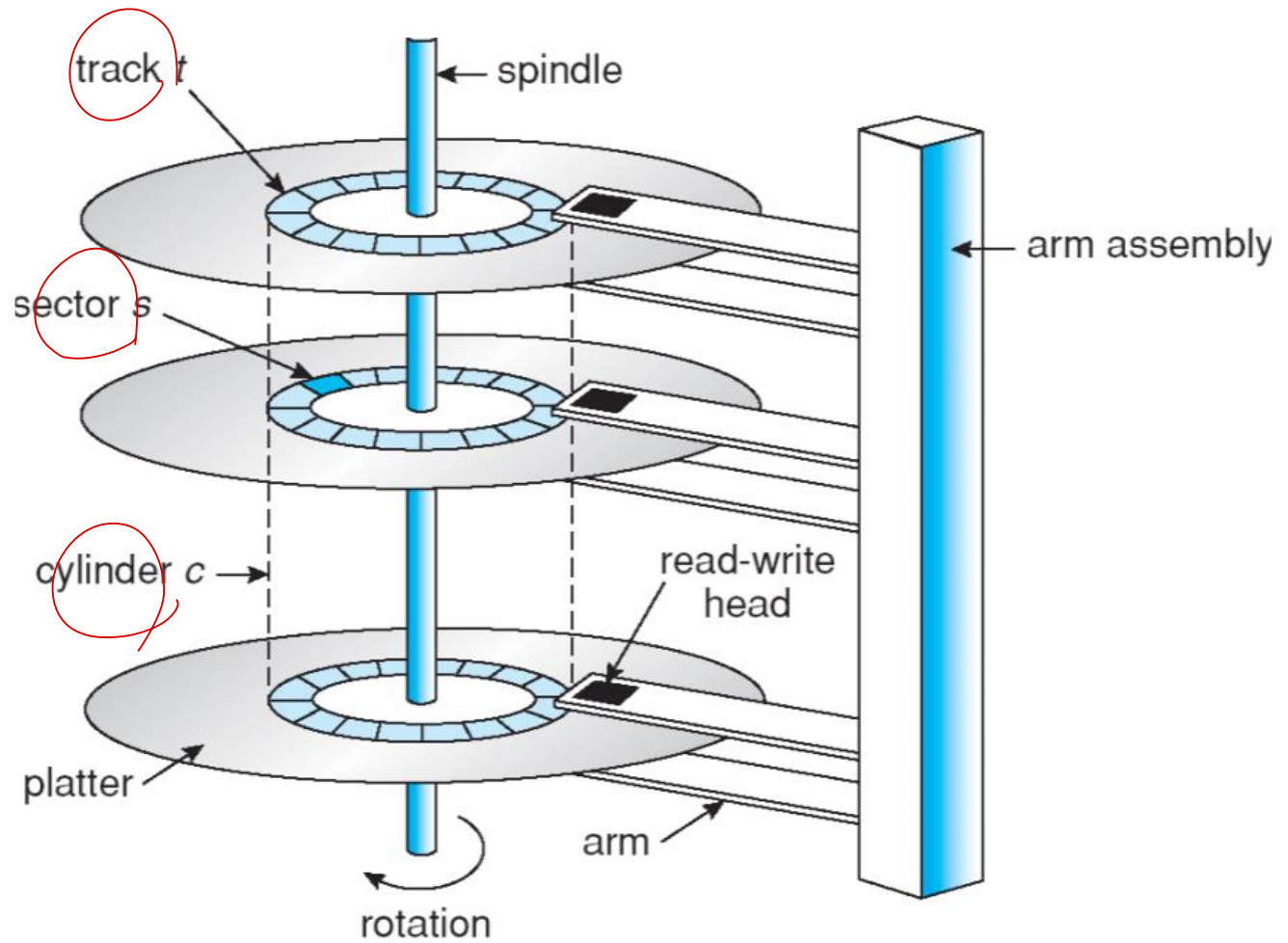


Storage Devices

- Magnetic disks
 - Storage that rarely becomes corrupted
 - Large capacity at low cost
 - Block level random access
 - Slow performance for random access
 - Better performance for streaming access
- Flash memory
 - Storage that rarely becomes corrupted
 - Capacity at intermediate cost (50x disk)
 - Block level random access
 -  – Good performance for reads; worse for random writes

Magnetic Disk

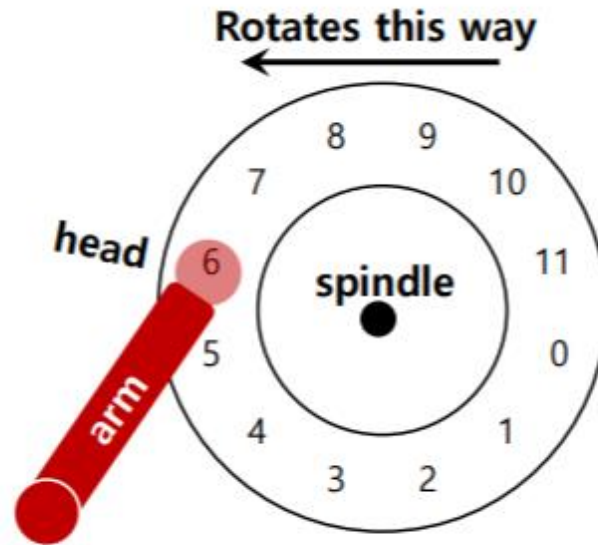




Basic Geometry

- Spindle
 - Spindle is connected to a motor that spins the platters around.
 - The rate of rotations is measured in RPM (Rotations Per Minute).
 - Typical modern values : 7,200 RPM to 15,000 RPM.
- Track
 - Concentric circles of sectors
 - Data is encoded on each surface in a track.
 - A single surface contains many thousands and thousands of tracks.
- Cylinder
 - A stack of tracks of fixed radius
 - Heads record and sense data along cylinders
 - Generally only one head active at a time

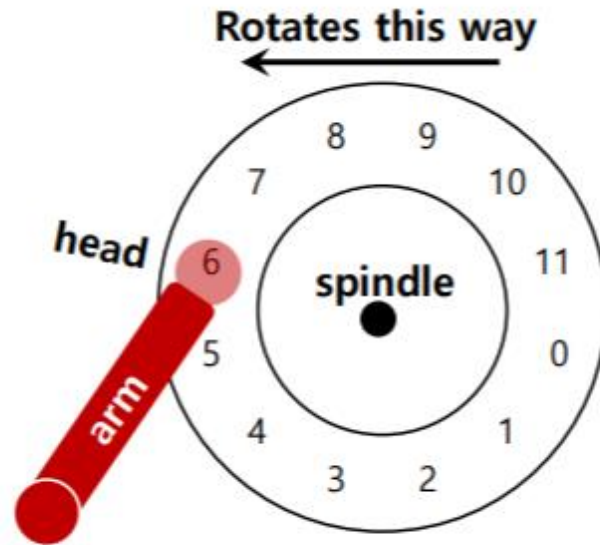
A Simple Disk Drive



A Single Track Plus A Head

- Disk head (One head per surface of the drive)
 - The process of reading and writing is accomplished by the disk head.
 - Attached to a single disk arm, which moves across the surface.

A Simple Disk Drive

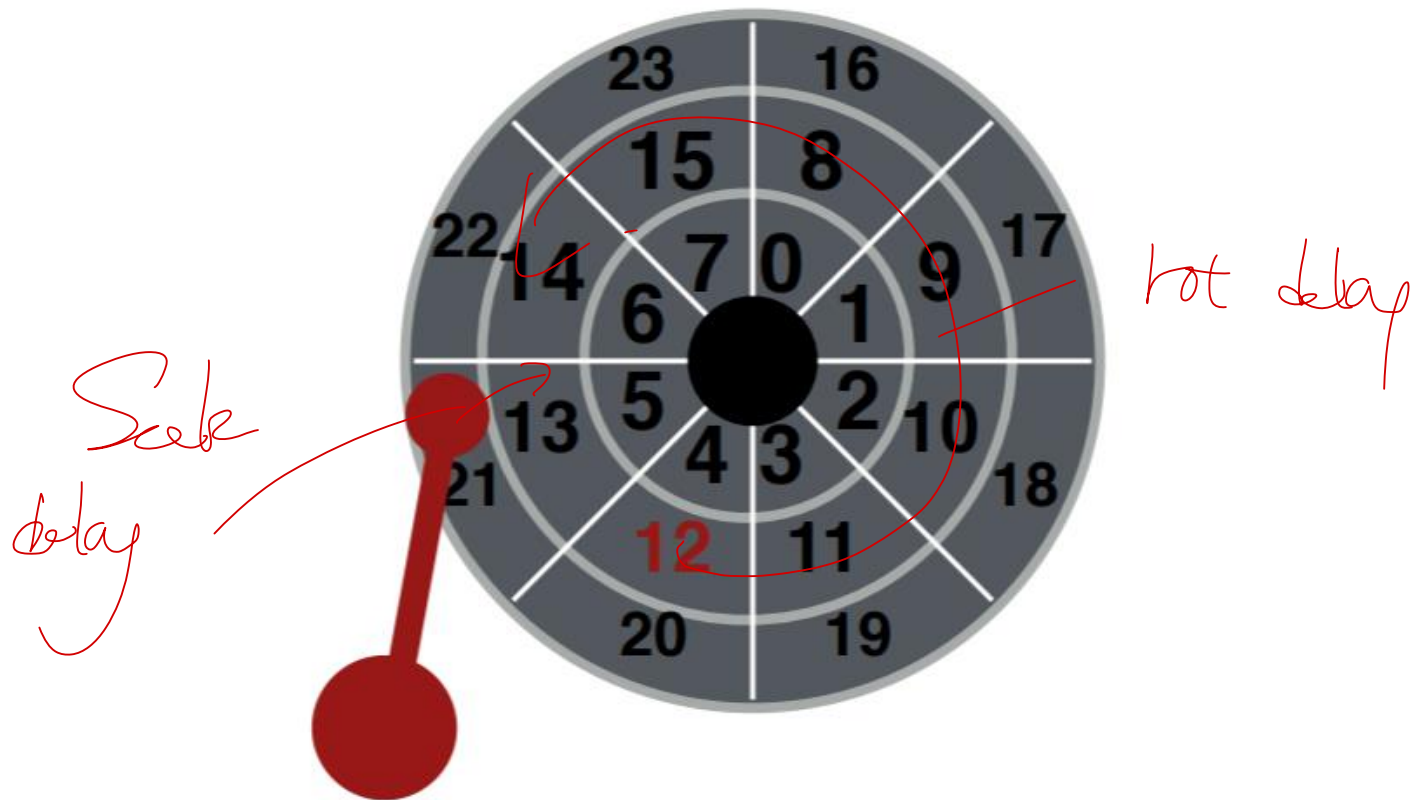


A Single Track Plus A Head

- Rotational delay: Time for the desired sector to rotate
 - Ex) Full rotational delay is R and we start at sector 6
 - Read sector 0: Rotational delay = $\frac{1}{12}R$
 - Read sector 5: Rotational delay = $\frac{11}{12}R$

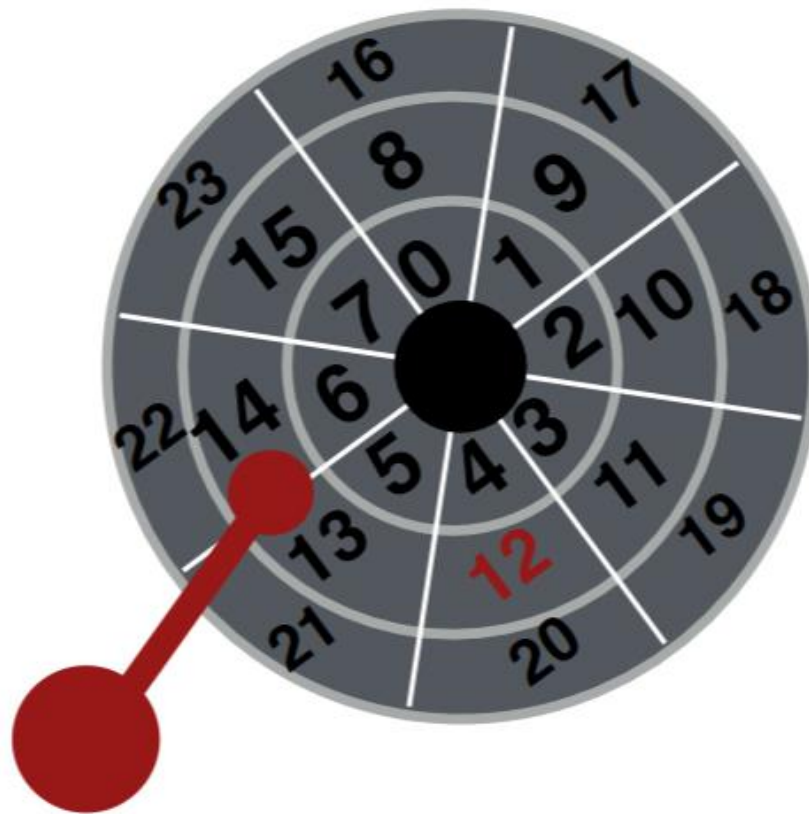
Multiple tracks

Read sector 12



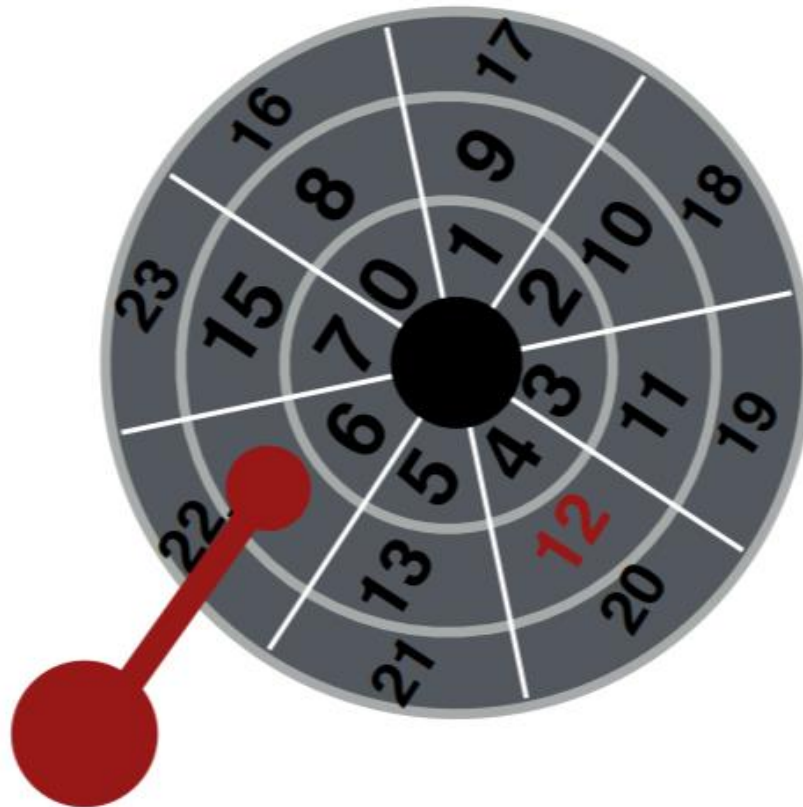
Multiple tracks

Read sector 12: Seek to next track



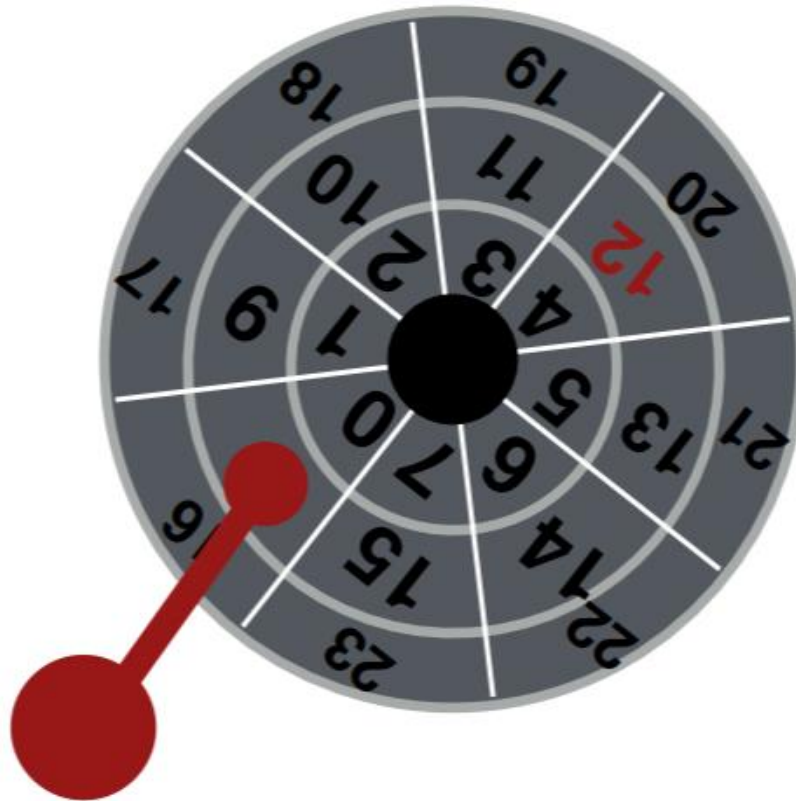
Multiple tracks

Read sector 12: wait for rotation



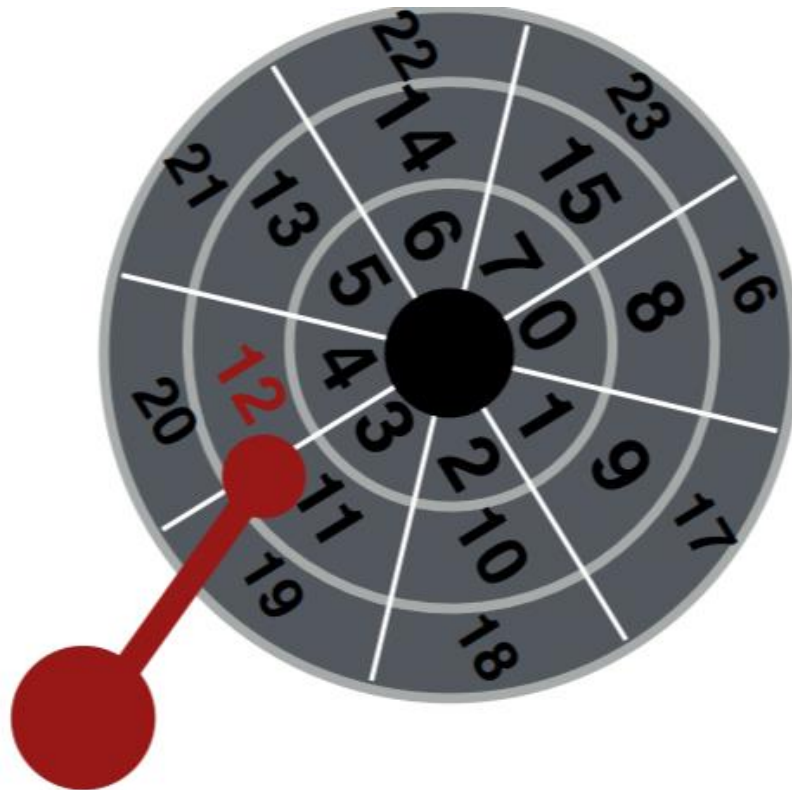
Multiple tracks

Read sector 12: wait for rotation



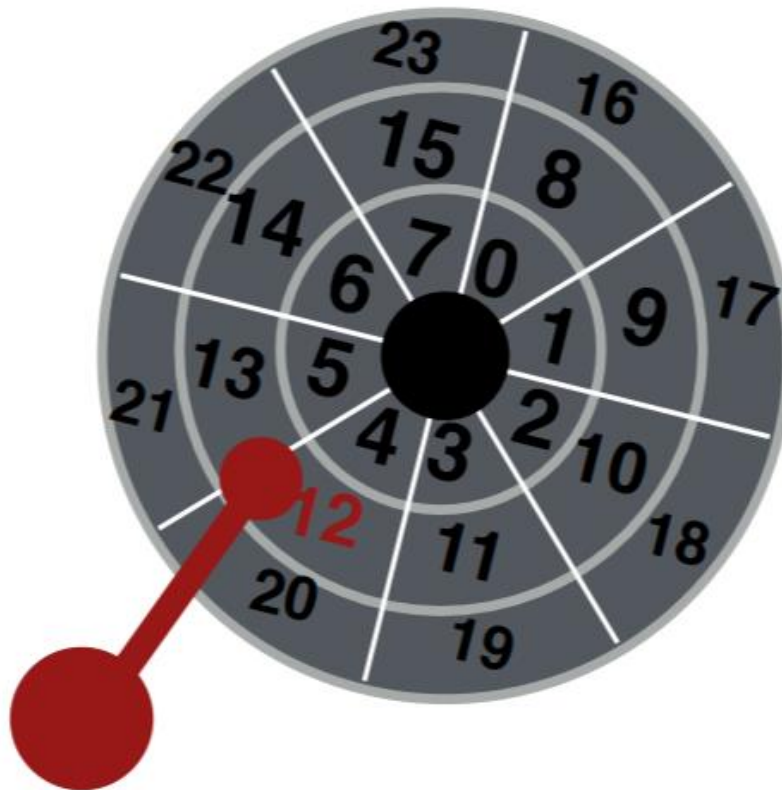
Multiple tracks

Read sector 12: wait for rotation

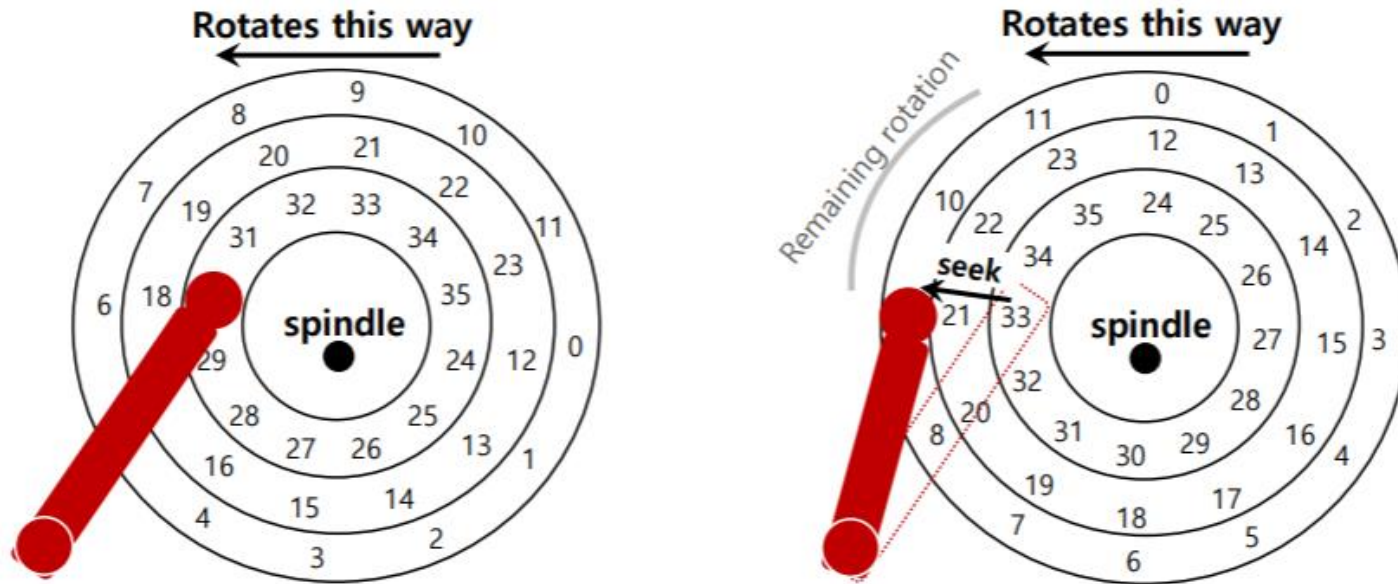


Multiple tracks

Read sector 12: Transfer data



Seek Time



- Seek: Move the disk arm to the correct track
 - Seek time: Time to move head to the track contain the desired sector.
 - ✱ – One of the most costly disk operations

Disk Performance

Disk Latency =

Seek Time + Rotation Time + Transfer Time

Seek Time: time to move disk arm over track (1-20ms)

Fine-grained position adjustment necessary for head to “settle”

Head switch time ~ track switch time (on modern disks)

Rotation Time: time to wait for disk to rotate under disk head

Disk rotation: 4 – 15ms (depending on price of disk)

On average, only need to wait half a rotation

Transfer Time: time to transfer data onto/off of disk

Disk head transfer rate: 50-100MB/s (5-10 usec/sector)

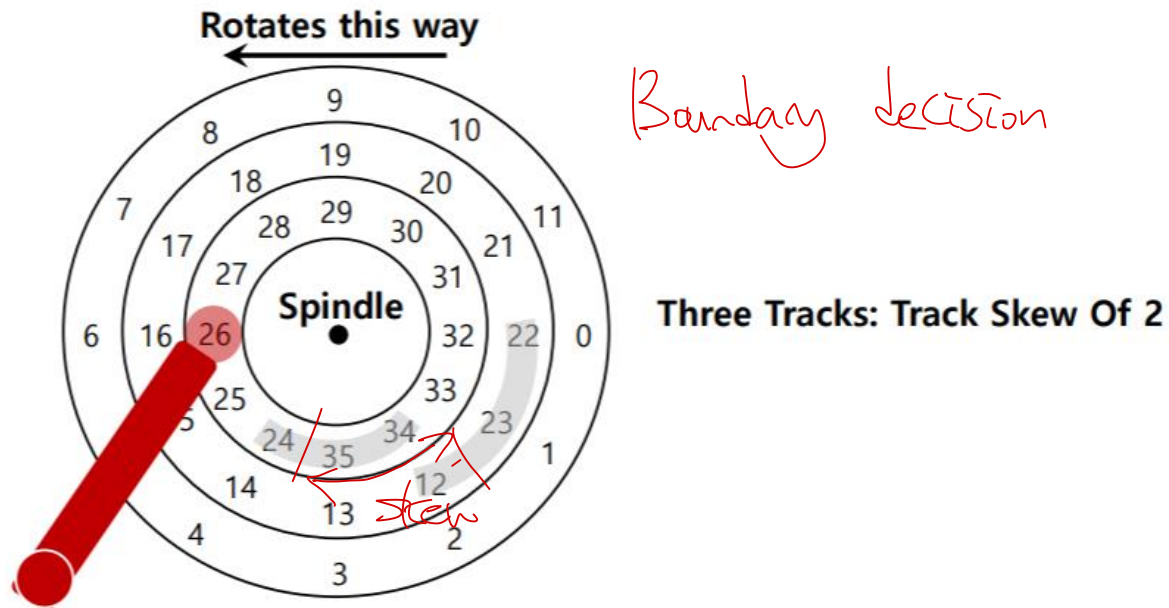
Host transfer rate dependent on I/O connector (USB, SATA, ...)

Seek

- Move head to specific track and keep it there - Resist physical shocks, imperfect tracks, etc.
- A seek consists of up to four phases:
 - speedup—accelerate arm to max speed or half way point
 - coast—at max speed (for long seeks)
 - slowdown—stops arm near destination
 - settle—adjusts head to actual desired track
- Seeks often take several milliseconds!
 - settling alone can take 0.5 to 2ms
 - entire seek often takes 1 - 20 ms

Track Skewing

- Track skewing
 - Sector numbers offset from one track to the next, to allow for disk head movement for sequential ops



Without track skew, the head would be moved to the next track but the desired next block would have already rotated under the head

Rotation

- Depends on rotations per minute (RPM)
 - 7200 RPM is common

$$\frac{1}{7200} \text{ min/rot}$$

- With 7200 RPM, how long to rotate around?
 - $1 / 7200 \text{ RPM} = 1 \text{ minute} / 7200 \text{ rotations} = 1 \text{ second} / 120 \text{ rotations} = 8.3 \text{ ms} / \text{rotation}$

- Average rotation time?

$$\downarrow \times \frac{1}{2}$$

$$4.15 \text{ ms}$$

Transfer

- The final phase of I/O
 - Data is either read from or written to the surface
- Pretty fast — depends on RPM and sector density
- 128 MB/s is typical for maximum transfer rate
- How long to transfer 512 B?

$$\frac{512}{128 \times 10^6} = 4 \mu s$$

4 μ s

Workload Access Pattern

- So...
 - seeks are super slow
 - rotations are slow
 - transfers are fast
- What kind of workload is fastest for disks?
 - Sequential: access sectors in order → Min seek delay & Rotation delay
 - Random: access sectors arbitrarily

Toshiba Disk (2008)

Size	
Platters/Heads	2/4
Capacity	320 GB
Performance	
Spindle speed	7200 RPM
Average seek time read/write	10.5 ms/ 12.0 ms
Maximum seek time	19 ms
Track-to-track seek time	1 ms
Transfer rate (surface to buffer)	54–128 MB/s
Transfer rate (buffer to host)	375 MB/s
Buffer memory	16 MB
Power	
Typical	16.35 W
Idle	11.68 W

Question

- How long to complete 500 (sectors) random disk reads, in FIFO order?
 - 1 Sector: 512 B
 - Seek: average 10.5 msec
 - Rotation: average 4.15 msec
 - Transfer (1 sector): 4 usec

$$500 (S + R + T)$$

$$14.65 \times 500 \text{ ms}$$

Question

- How long to complete 500 random disk reads, in FIFO order?
 - Seek: average 10.5 msec
 - Rotation: average 4.15 msec
 - Transfer (1 sector): 4 usec
- $500 * (10.5 + 4.15 + 0.004)/1000 = 7.3$ seconds

Question

- How long to complete 500 sequential disk reads?

↳ 1 seek & rot

- 1 Sector: 512 B
- Seek: average 10.5 msec
- Rotation: average 4.15 msec
- Transfer (1 sector): 4 usec

$S + R + 500T$

4×500

$2000 \mu s \rightarrow 2ms$

Question

- How long to complete 500 sequential disk reads?

- Seek Time: 10.5 ms (to reach first sector)
- Rotation Time: 4.15 ms (to reach first sector)
- Transfer Time:

$$500 \text{ sectors} * 512 \text{ bytes} / 128\text{MB/sec} = 2\text{ms}$$

first sector

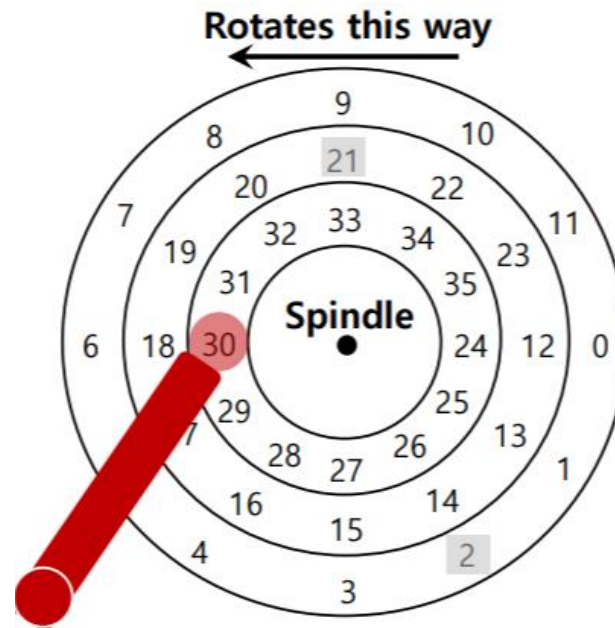
$$\text{Total: } 10.5 + 4.15 + 2 = 16.7 \text{ ms}$$

c.f. Complete Random 500 sector: 7.3 sec (437x)

→ Sequential is much faster

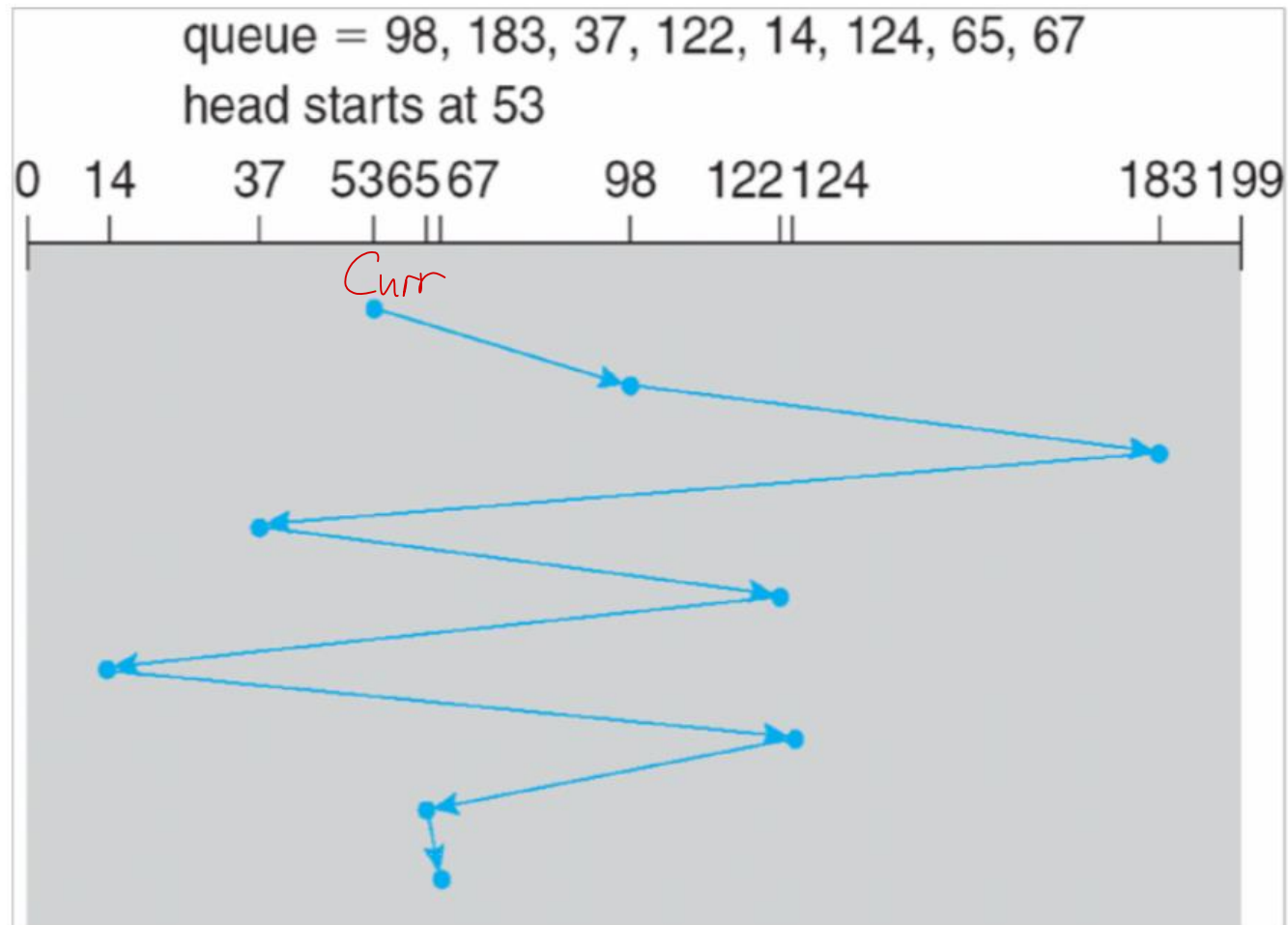
Disk Scheduling

- Disk Scheduler decides which I/O request to schedule next



FCFS Example

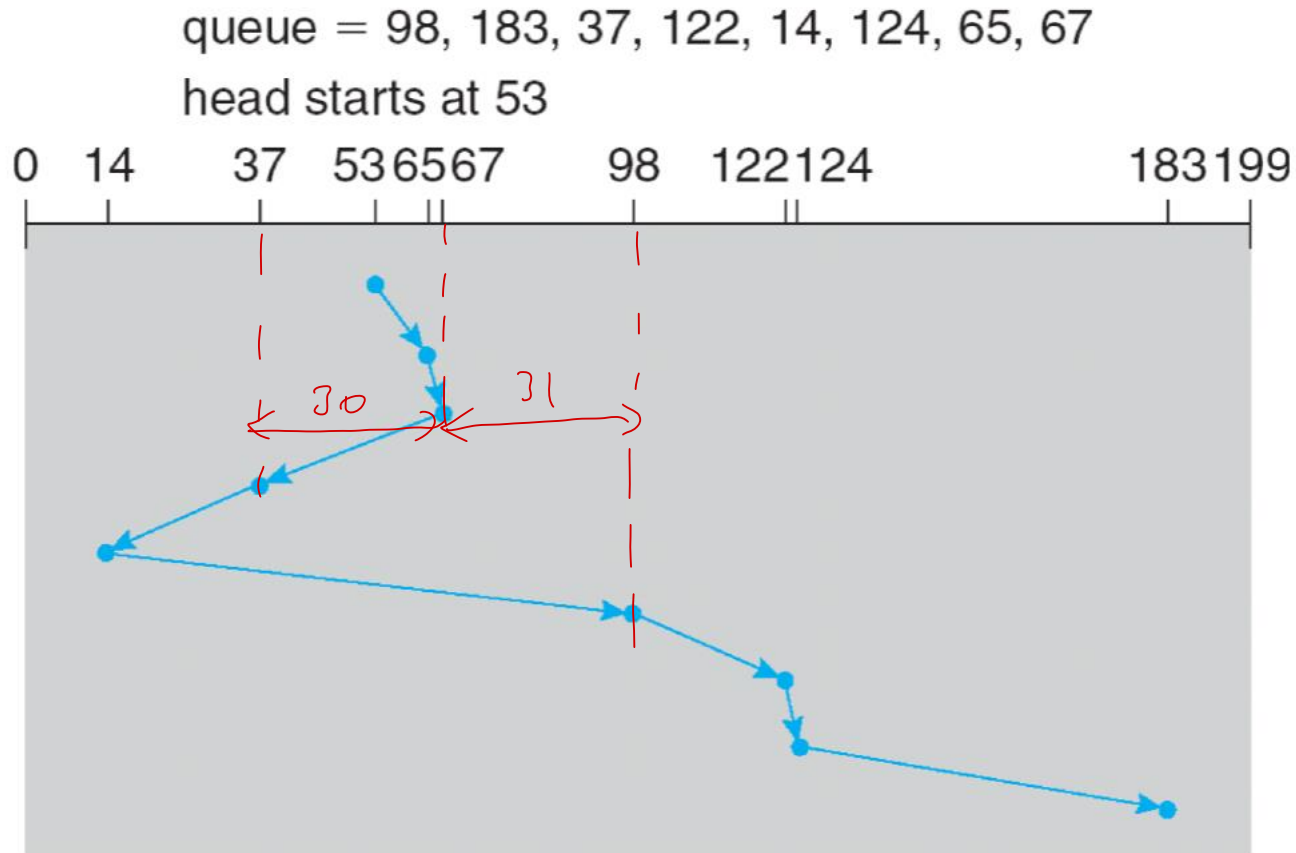
FIFO




FCFS

- “First Come First Served”
 - Process disk requests in the order they are received
- Advantages
 - Easy to implement
 - Good fairness
- Disadvantages
 - ✱ – Cannot exploit request locality
 - Slow: Increases average latency, decreasing throughput

SSTF (Shortest Seek Time First)

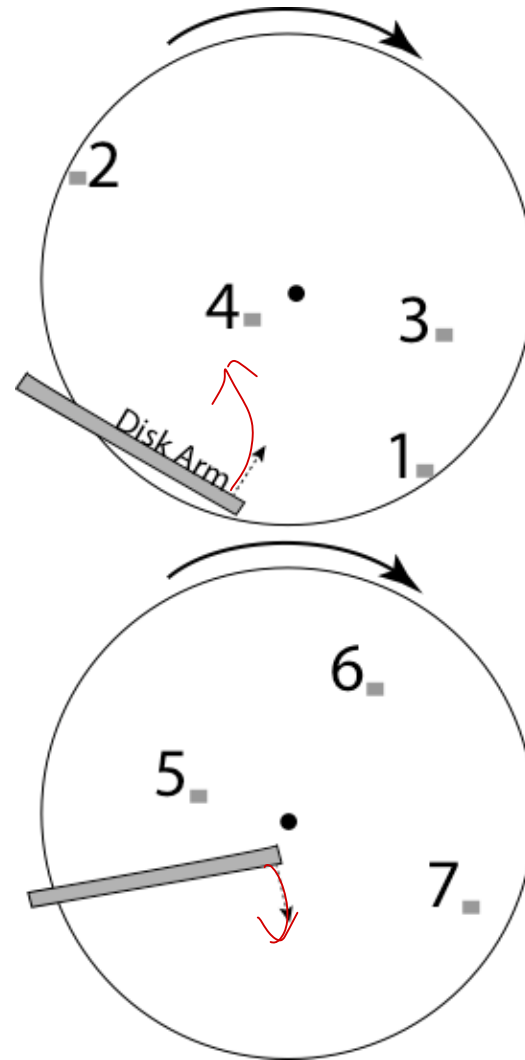


SSTF (Shortest Seek Time First)

- Order the queue of I/O request by track
 - Pick requests on the nearest track to complete first
 - Also called shortest positioning time first (SPTF)
- Advantages
 - Exploits locality of disk requests
 - Higher throughput
- Disadvantages
 -  – Starvation
 - May not be optimal

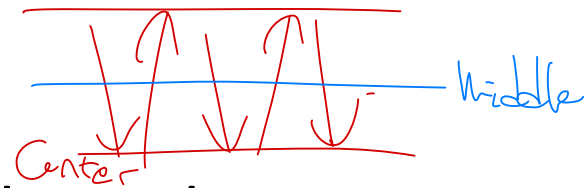
Disk Scheduling

- SCAN: move disk arm in one direction, until all requests satisfied, then reverse direction
- Also called “elevator scheduling”

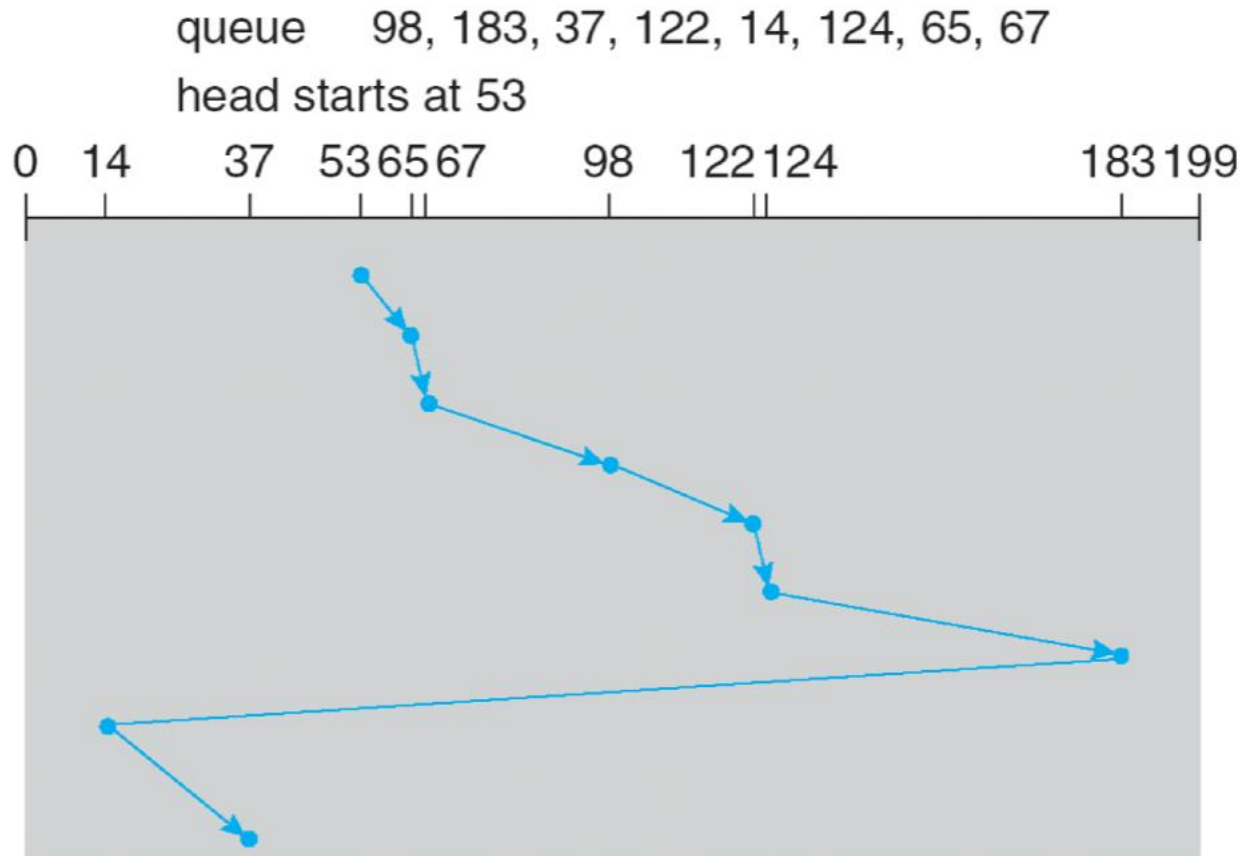


Elevator Scheduling (SCAN)

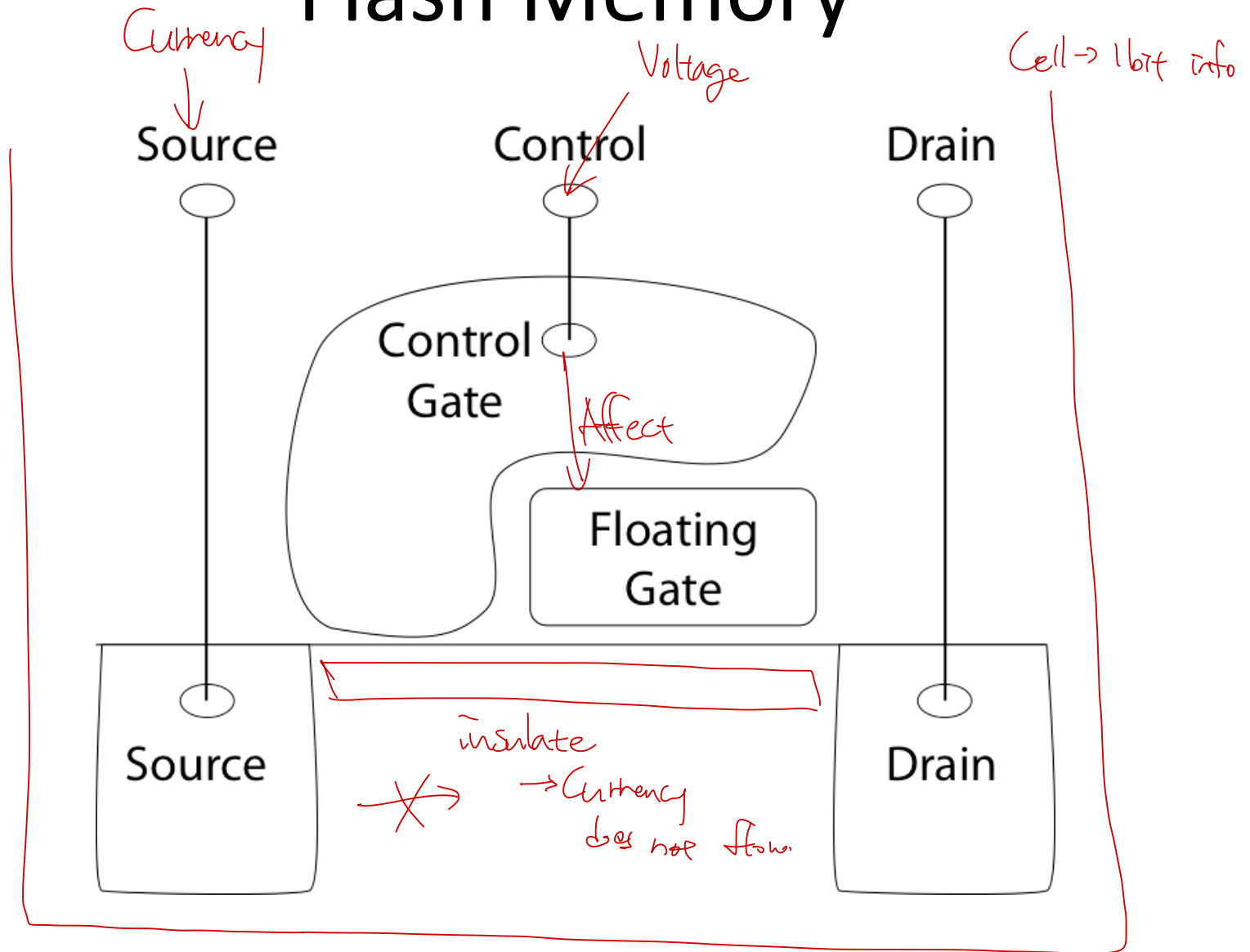
- Advantages
 - Takes advantage of locality
 - Bounded waiting
- Disadvantages
 - Unfair: Cylinders in the middle get better service
 - Might miss locality SSTF could exploit
- CSCAN: Only sweep in one direction
 - Very commonly used algorithm in Unix



Elevator Scheduling (C-SCAN)



Flash Memory



Flash Memory

- Writes must be to “clean” cells; no update in place → Erase before write
 - Large block erasure required before write
 - Erasure block: 128 – 512 KB → old Now: 64MB to 1GB
 - Erasure time: Several milliseconds → Slow
- Write/read page (2-4KB)
 - 50-100 usec

Flash is faster than hard drives

Random 4K read

Cheaper / Capacity
↗

Hard Drives

PCIe-Flash
2007



Latency

Lat.: 7.1ms

much faster

BW: 2.6MB/s

68us

250MB/s

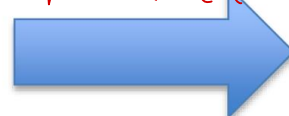
Bandwidth

1x

104x

1x

96x

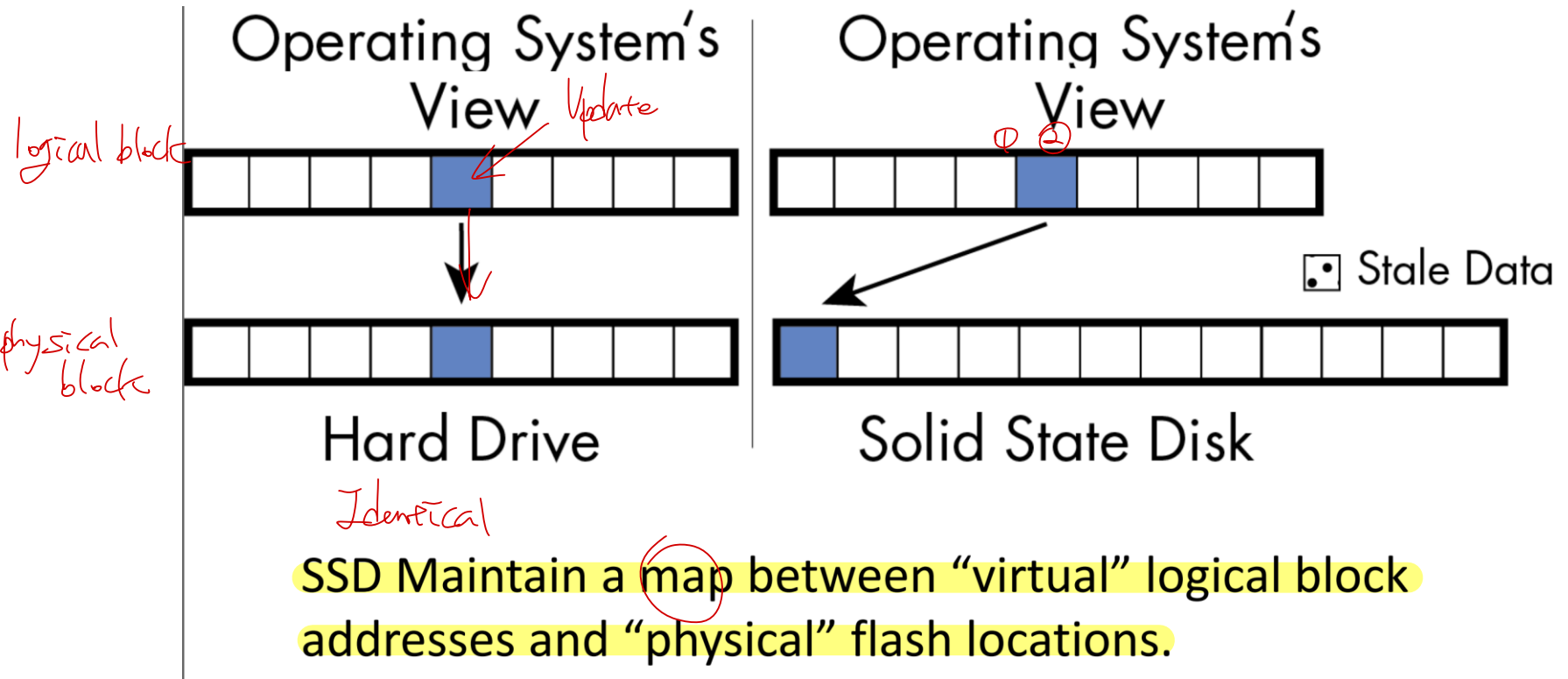


Question

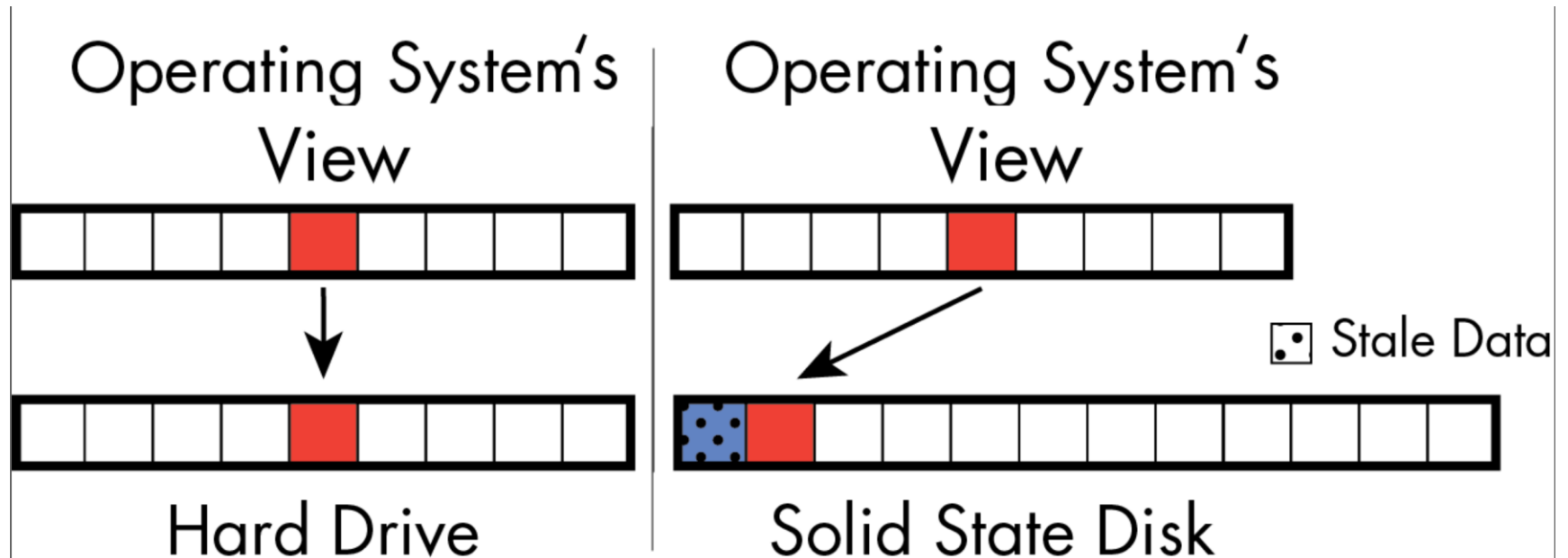
- Why are random writes so slow?
 - Random write: 2000/sec → Needs erase operation first.
 - Random read: 38500/sec

Sequential write is much better

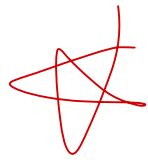
Writing data



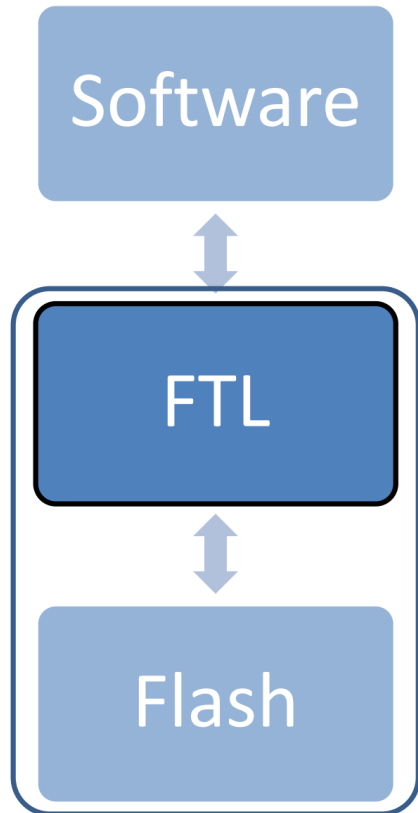
Updating data



When you overwrite data, it goes to a new location.



Flash Translation Layer



User

- Logical Block Address

Flash

- Write pages in order
- Erase/Write granularity 남아뭇야.
- Wears out → FTL 쓰기 작업은 점점 "느린".

FTL

- Logical → Physical map
- Wear leveling
- Power cycle recovery

Erase before write

≡ physical block

Page 0	Page 1	Page 2	Page 3
00011000	11001110	00000001	00111111
VALID	VALID	VALID	VALID

e.g., Erasure Block size = 4 pages

Write "00000011" to page 0

↓ *Erase*

Page 0	Page 1	Page 2	Page 3
11111111	11111111	11111111	11111111
ERASED	ERASED	ERASED	ERASED

write page 0

↓

Page 0	Page 1	Page 2	Page 3
00000011	11111111	11111111	11111111
VALID	ERASED	ERASED	ERASED

File System – Flash

- How does Flash device know which blocks are live?
 - Live blocks must be remapped to a new location during erasure
- TRIM command
 - File system tells device when blocks are no longer in use

File System Abstraction (will discuss)

- File system
 - Persistent, named data
 - Hierarchical organization (directories, subdirectories)
 - Access control on data
- File: named collection of data
 - Linear sequence of bytes (or a set of sequences)
 - Read/write or memory mapped
- Crash and storage error tolerance
 - Operating system crashes (and disk errors) leave file system in a valid state
- Performance
 - Achieve close to the hardware limit in the average case