CS304 Operating Systems

DR GAYATHRI ANANTHANARAYANAN

gayathri@iitdh.ac.in

Materials in these slides have been borrowed from textbooks and existing operating systems courses

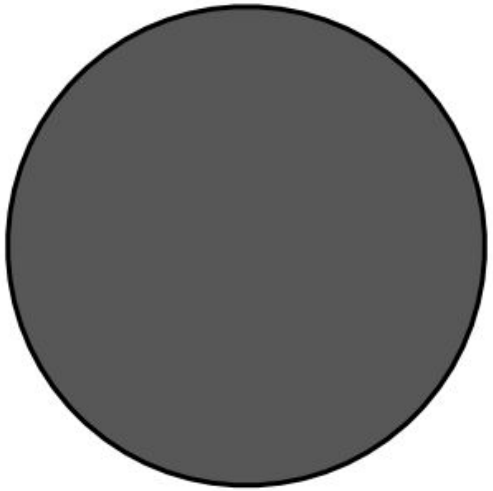
Hard Disk – Basic Interface

- Interface: a set of 512-byte blocks (sectors),that can be read or written atomically [sectors are numbered from 0 to n−1 on a disk with n sectors]
- Disk has a sector-addressable address space
  -Appears as an array of sectors.
- Main operations: reads + writes to sectors
- Mechanical (slow) nature makes management “interesting”

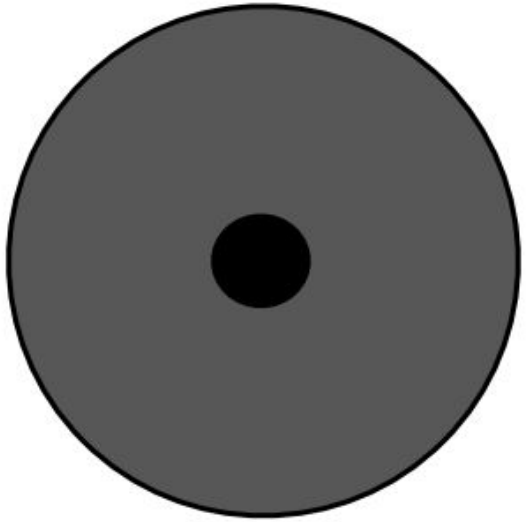
Internal Structure

- Internals: one or more platters, connected by a spindle, spinning at ~10K RPM (rotations per minute)
- Each platter has a disk head and arm
- A platter is divided into multiple tracks, and each track into 512-byte sectors

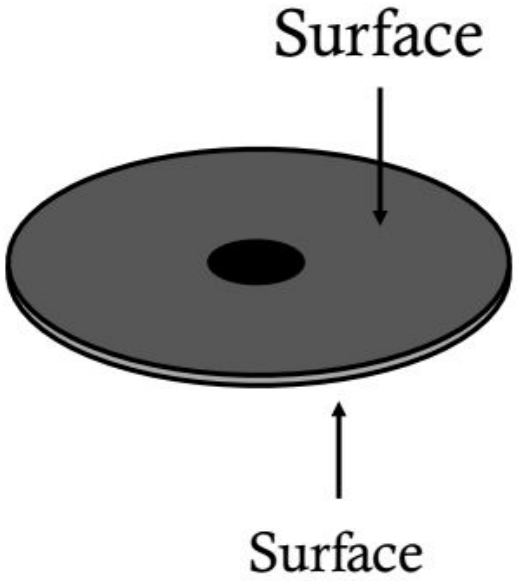
Internal Structure

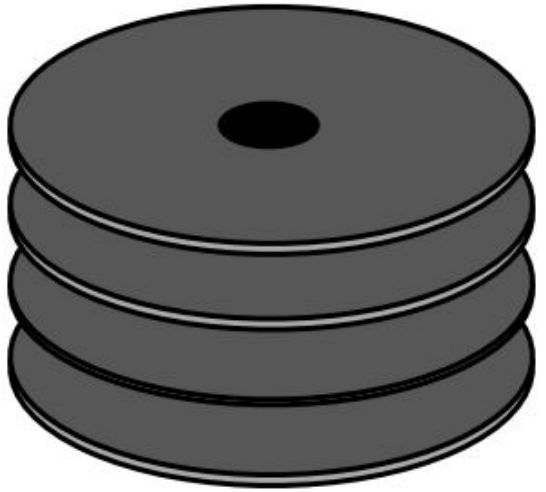


Platter [covered with a magnetic film] [Aluminum coated with a thin magnetic layer]



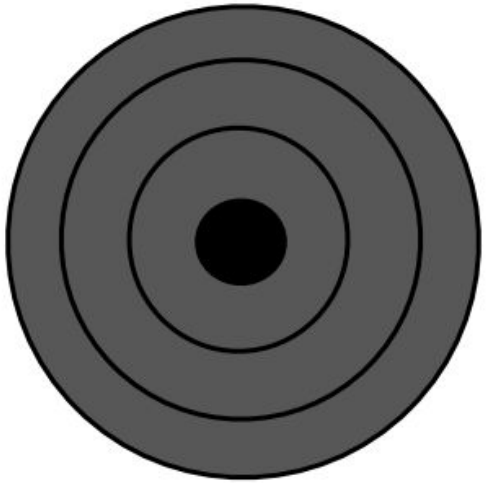
Spindle





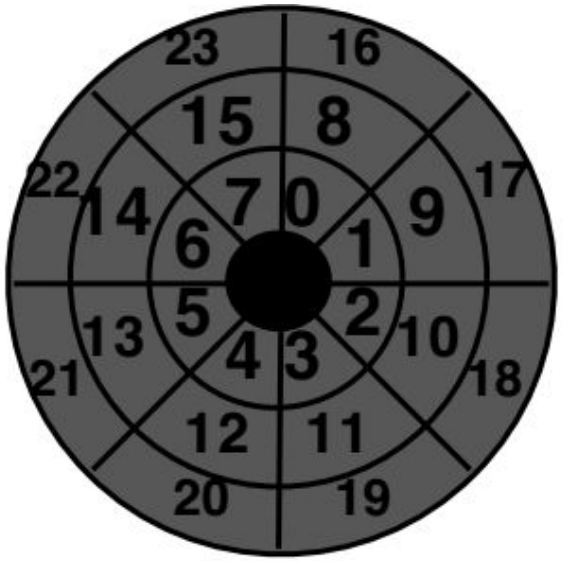
Many platters may be bound to the spindle

Internal Structure

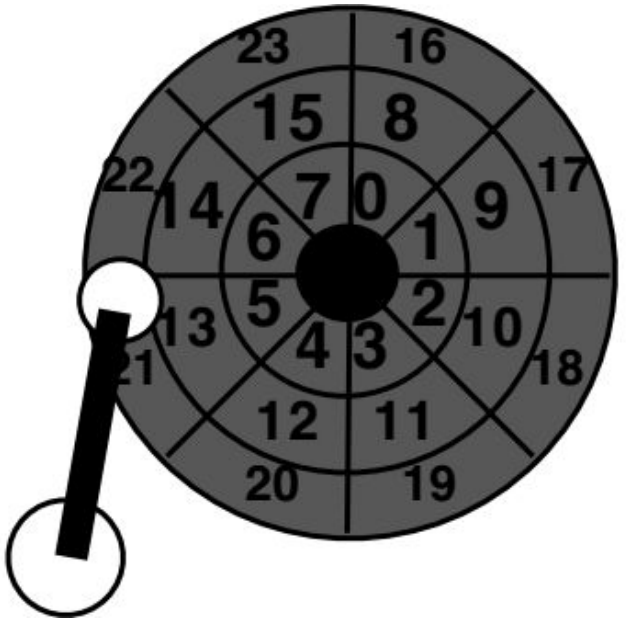


Each surface is divided into rings called tracks.

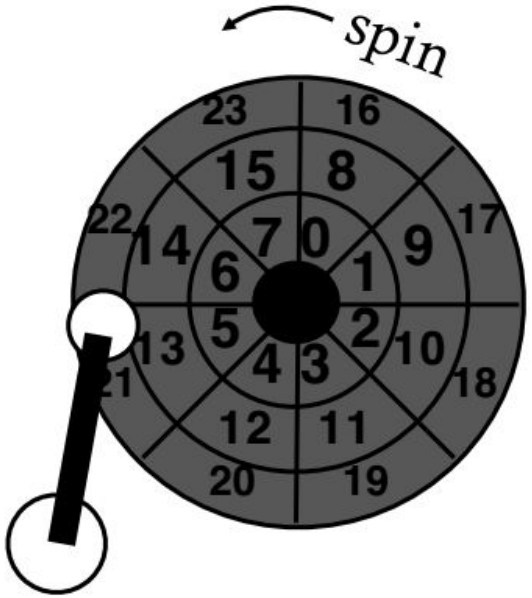
A stack of tracks (across platters) is called a cylinder.



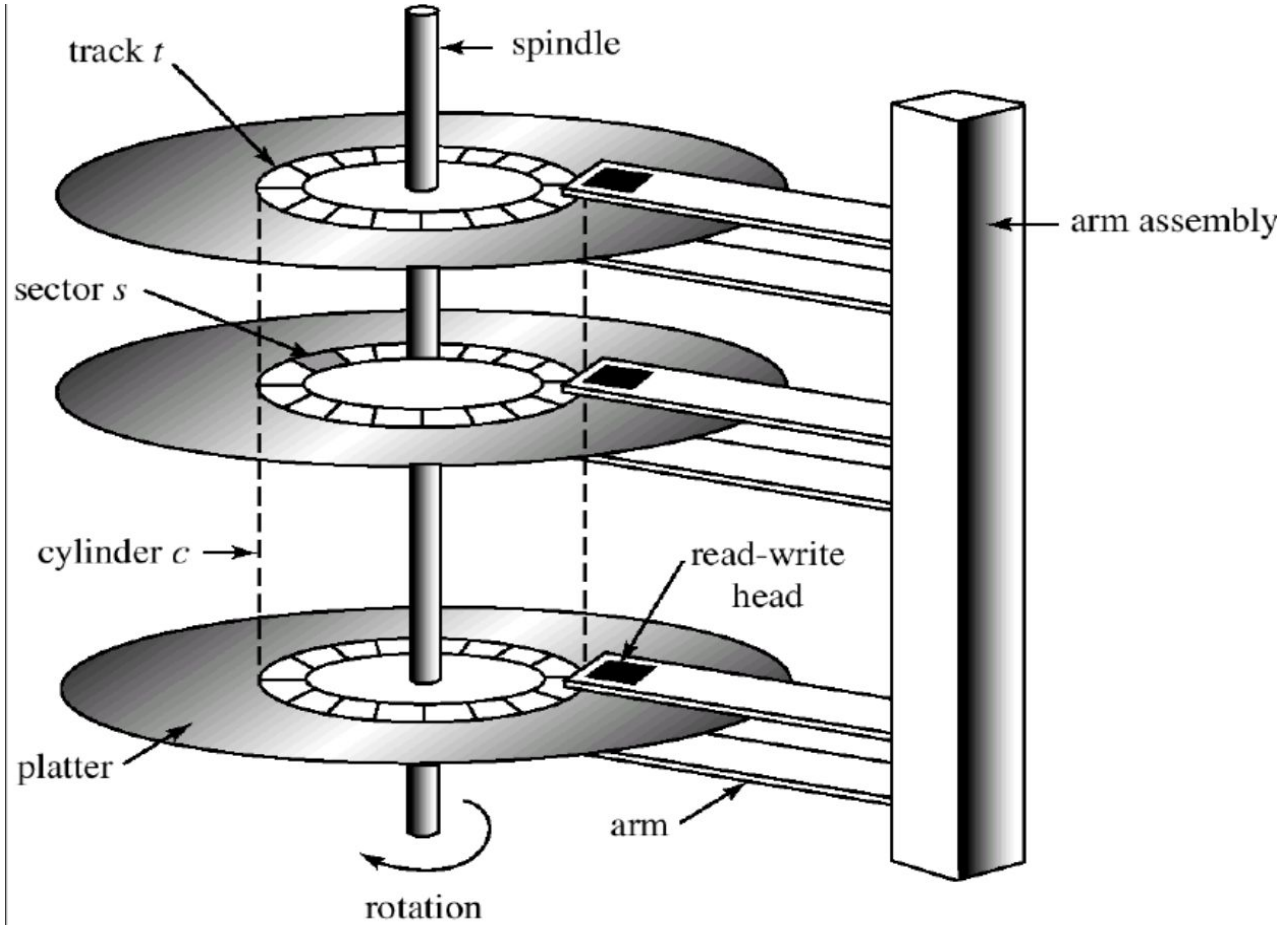
The tracks are divided into numbered sectors.



Heads on a moving arm can read from each surface.



Spindle/platters rapidly spin



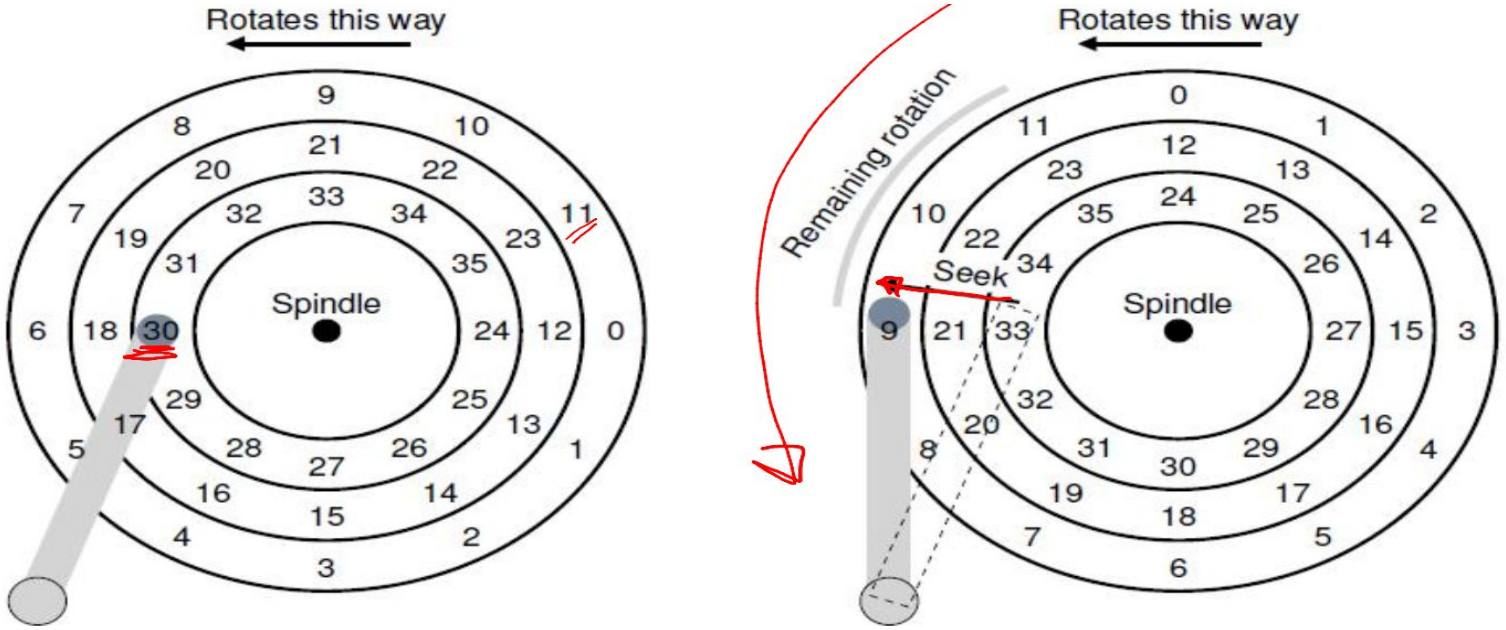
Hard Disk Demo

https://www.youtube.com/watch?v=9eMWG3fwiEU&t=30s

https://www.youtube.com/watch?v=L0nbo1VOF4M

What happens when accessing a sector?

- Suppose disk head at 30, need to access 11
- Seek to the correct track, wait for disk to rotate



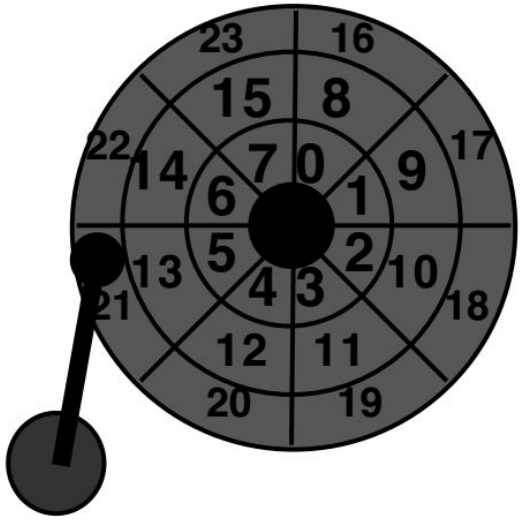
Positioning

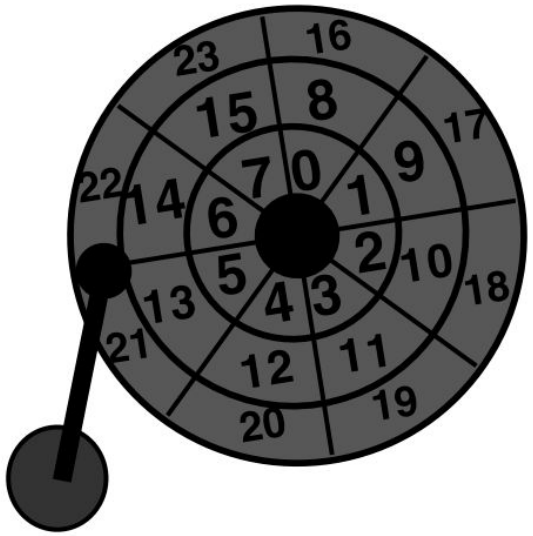
Drive servo system keeps head on track

- How does the disk head know where it is?
- Platters not perfectly aligned, tracks not perfectly concentric (runout) --difficult to stay on track
- More difficult as density of disk increase
  -More bits per inch (BPI), more tracks per inch (TPI)

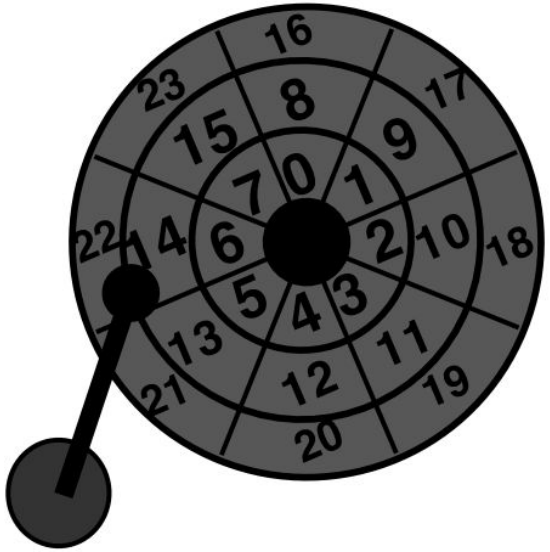
Use servo burst:

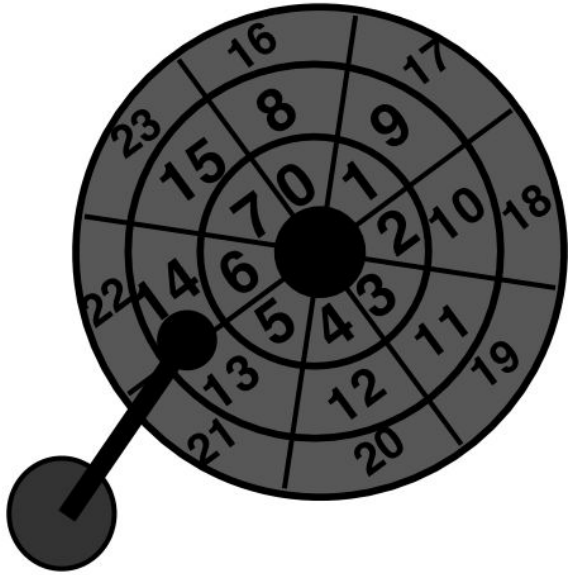
- Record placement information every few (3-5) sectors
- When head cross servo burst, figure out location and adjust as needed



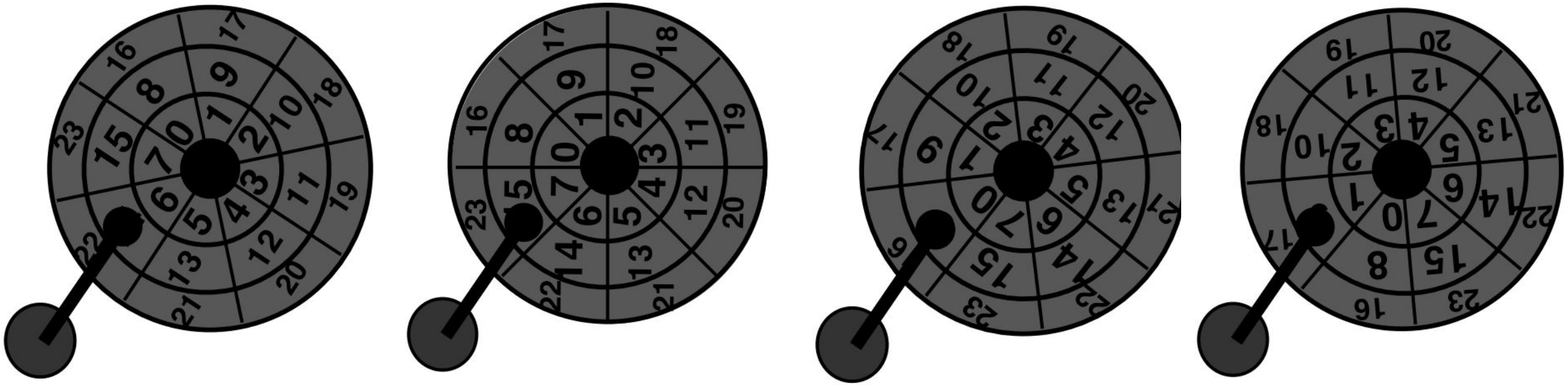


Let’s Read 12!

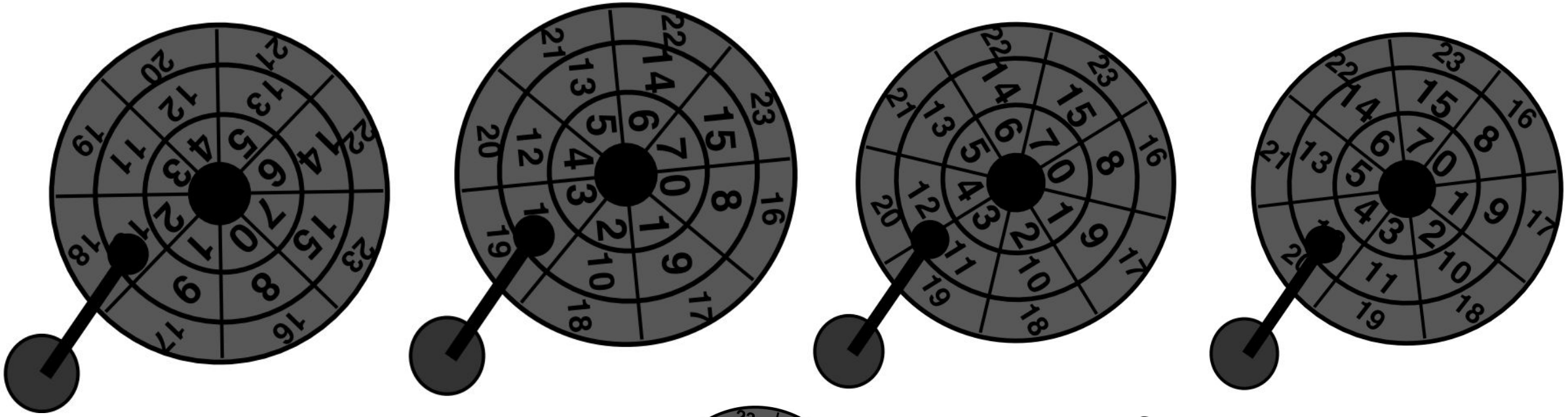




Seek to right track



Wait for Rotation



Wait for Rotation

Transfer data

Time to Read/write[I/O Operation]

- Three components: Time = seek + rotation + transfer time
- Seek cost: Function of cylinder distance -- Not purely linear cost
- Must accelerate, coast, decelerate, settle
- Settling alone can take 0.5 -2 ms
- Entire seeks often takes several milliseconds -- 4 -10 ms
- Approximate average seek distance = 1/3 max seek distance

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]

Acceleration --> Coasting --> Deceleration --> Settling

- Acceleration: The disk arm gets moving.
- Coasting: The arm is moving at full speed.
- Deceleration: The arm slows down.
- Settling: The head is carefully positioned over the correct track.
- The settling time is often quite significant, e.g., 0.5 to 2ms.

Time to Read/write[I/O Operation]

Depends on rotations per minute (RPM)

- 7200 RPM is common, 15000 RPM is high end.

With 7200 RPM, how long to rotate around?

1 / 7200 RPM =1 minute / 7200 rotations = 1 second / 120 rotations =8.3ms / rotation

Average rotation? 8.3 ms / 2 = 4.15 ms

Time to Read/write[I/O Operation]

Pretty fast —depends on RPM and sector density.

100+ MB/s is typical for maximum transfer rate

How long to transfer 512-bytes?

512 bytes * (1s / 100 MB)= 5 us

Workload Performance

-seeks are slow

-rotations are slow

-transfers are fast

What kind of workload is fastest for disks?

Sequential: access sectors in order (transfer dominated)

Random: access sectors arbitrarily (seek + rotation dominated)

Workload Performance

- I/O time ($T_{I/O}$): $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

- The rate of I/O ($R_{I/O}$): $R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$

	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

Sequential workload: what is throughput for each?

Cheetah: 125 MB/s.
Barracuda: 105 MB/s.

Workload Performance

Random workload: what is throughput for each?

(what else do you need to know?)

Workload Performance

What is size of each random read? **Assume 16-KB reads**

How long does an average random 16-KB read take w/ Cheetah?

Seek + rotation + transfer;

Seek = 4 ms

Average rotation in ms?

$$\text{avg rotation} = \frac{1}{2} \times \frac{1\ \text{min}}{15000} \times \frac{60\ \text{sec}}{1\ \text{min}} \times \frac{1000\ \text{ms}}{1\ \text{sec}} = 2\ \text{ms}$$

Workload Performance

Transfer of 16KB ?

$$\text{transfer} = \frac{1\ \text{sec}}{125\ \text{MB}} \times 16\ \text{KB} \times \frac{1{,}000{,}000\ \text{us}}{1\ \text{sec}} = 125\ \text{us}$$

$$\text{Cheetah time} = 4\text{ms} + 2\text{ms} + 125\text{us} = 6.1\text{ms}$$

$$\text{throughput} = \frac{16\ \text{KB}}{6.1\text{ms}} \times \frac{1\ \text{MB}}{1024\ \text{KB}} \times \frac{100\ \text{ms}}{1\ \text{sec}} = 2.5\ \text{MB/s}$$

Workload Performance

How long does an average random 16-KB read take w/ Barracuda?

$$\text{Barracuda time} = 9\text{ms} + 4.1\text{ms} + 149\text{us} = 13.2\text{ms}$$

$$\text{Time} = \text{seek} + \text{rotation} + \text{transfer}$$

$$\text{Seek} = 9\text{ms}$$

$$\text{avg rotation} = \frac{1}{2} \times \frac{1\text{ min}}{7200} \times \frac{60\text{ sec}}{1\text{ min}} \times \frac{1000\text{ ms}}{1\text{ sec}} = 4.1\text{ ms}$$

$$\text{throughput} = \frac{16\text{ KB}}{13.2\text{ms}} \times \frac{1\text{ MB}}{1024\text{ KB}} \times \frac{1000\text{ ms}}{1\text{ sec}} = 1.2\text{ MB/s}$$

$$\text{transfer} = \frac{1\text{ sec}}{105\text{ MB}} \times 16\text{ KB} \times \frac{1{,}000{,}000\text{ us}}{1\text{ sec}} = 149\text{ us}$$

Time taken for I/O Operation

- Time taken to read/write a block consists of
  - Seek time to get to the right track (few ms)
  - Rotational latency for disk to spin to correct sector on the track (few ms)
  - Transfer time to read sector (few tens microsec)
- Given high seek and rotational latency, usually rate of sequential access > rate of random access

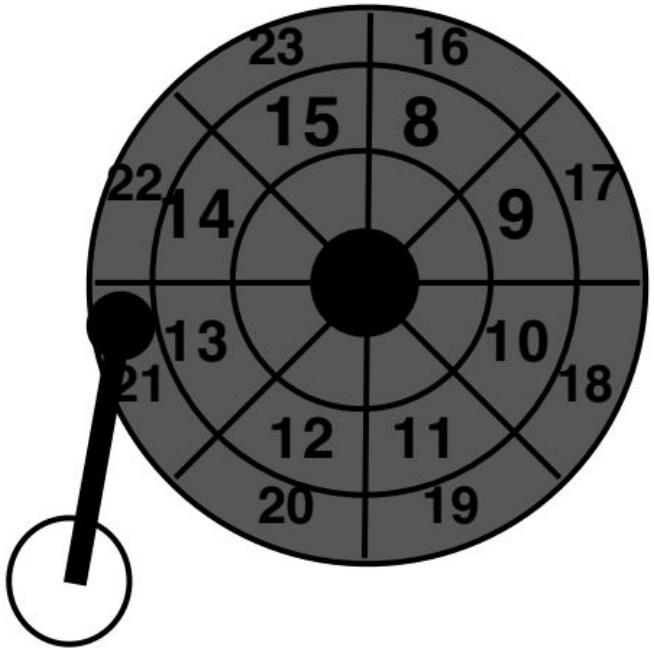
	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

Other improvements

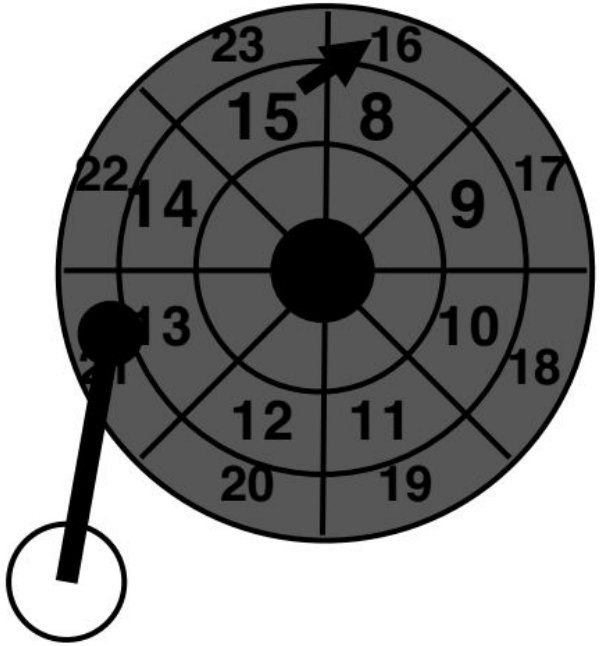
- Track Skew
- Zones
- Cache

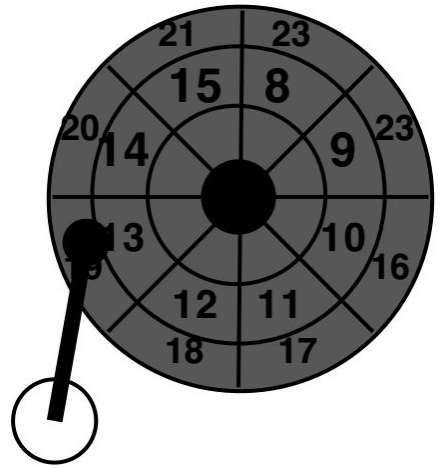
Track Skew

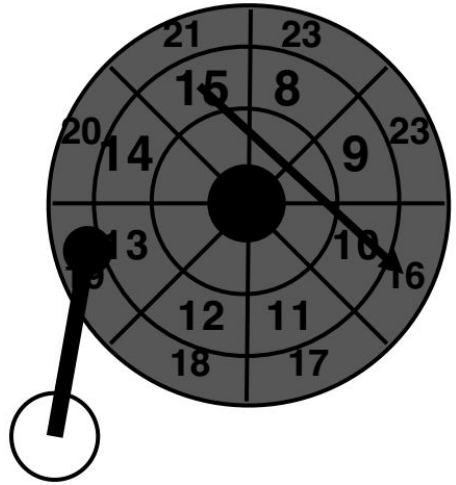
Imagine sequential reading,
how should sectors numbers be laid out on disk?



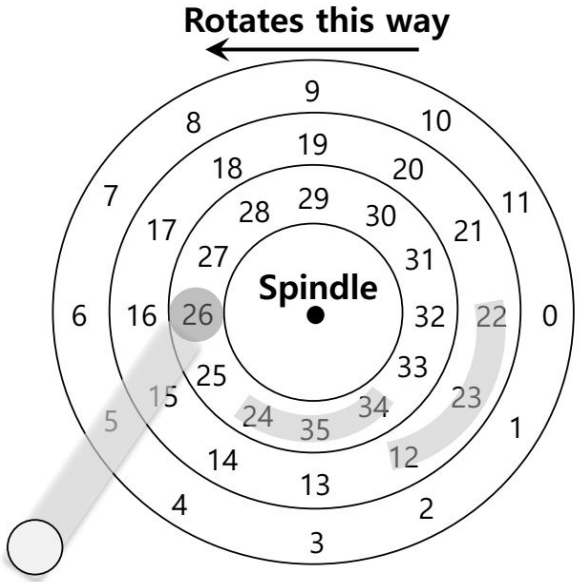
When reading 16 after 15, the head won't settle quick enough, so we need to do a rotation.







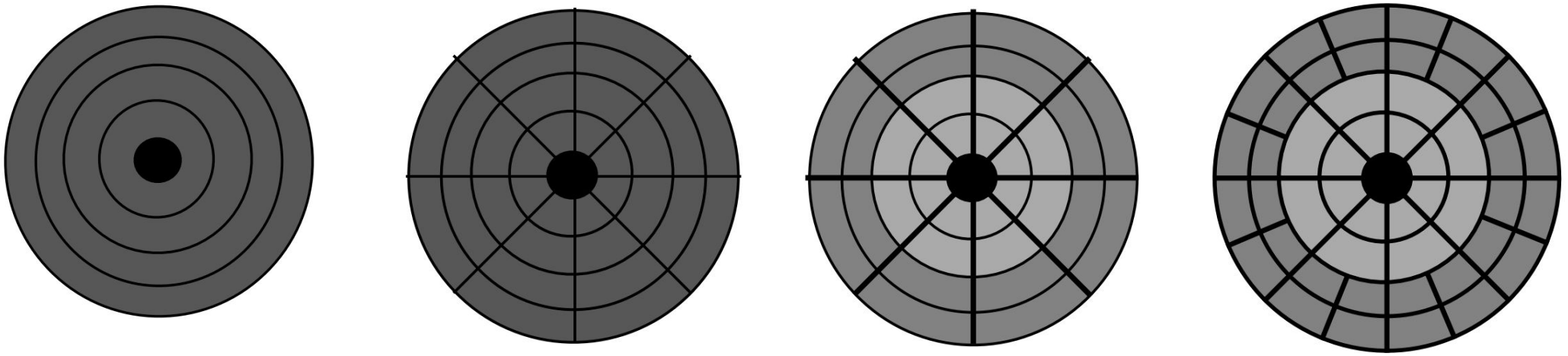
- Make sure that sequential reads can be properly serviced **even when crossing track boundaries**.



Three Tracks: Track Skew Of 2

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

Zones



ZBR (Zoned bit recording):
More sectors on outer tracks

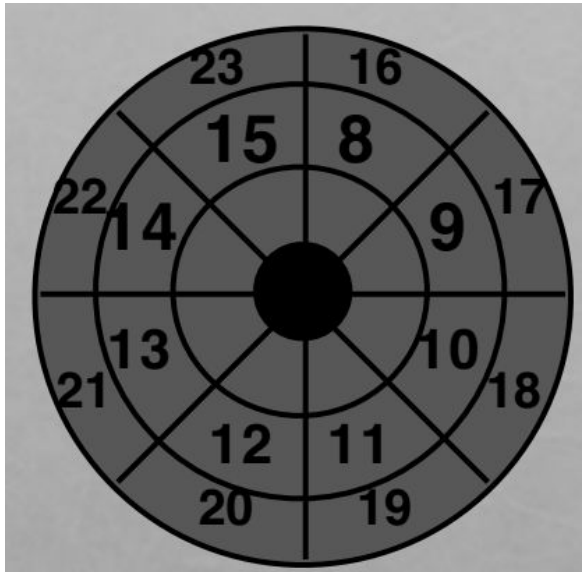
Drive Cache

Drives may cache both reads and writes.

- OS caches data too

What advantage does caching in drive have for reads?

What advantage does caching in drive have for writes?



Buffering

- Disks contain internal memory (2MB-16MB) used as cache
  - Read-ahead: “Track buffer”-- Read contents of entire track into memory during rotational delay
- Write caching with volatile memory
  - Immediate reporting[WB]: Claim written to disk when not
  - Data could be lost on power failure
- Tagged command queueing
  - Have multiple outstanding requests to the disk
  - Disk can reorder (schedule) requests for better performance

Disk Scheduling

- Given a stream of I/O requests, in what order should they be served?
- Much different than CPU scheduling
- Position of disk head relative to request position matters more than length of job

First Come First Served

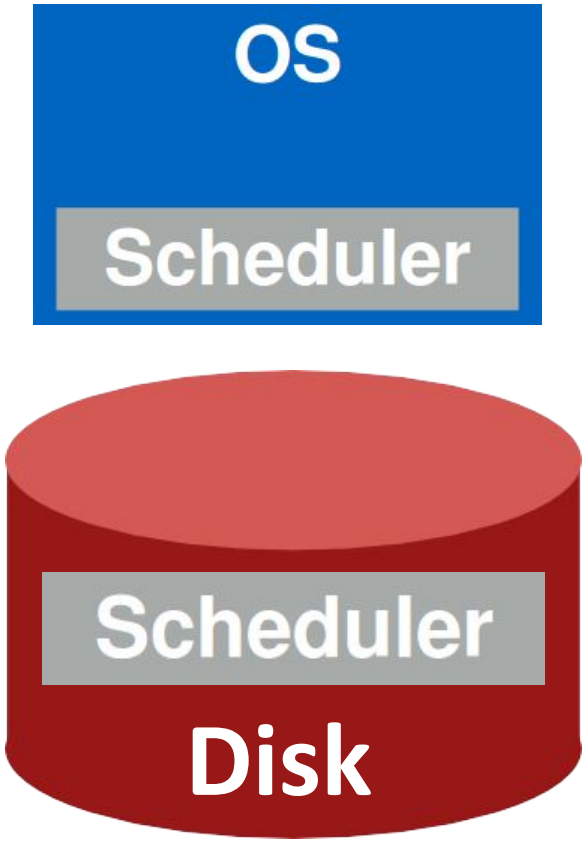
Assume seek+rotate = 10 ms for random request

How long (roughly) does the below workload take?

- Requests are given in sector numbers

300001, 700001, 300002, 700002, 300003, 700003 ~60ms

300001, 300002, 300003, 700001, 700002, 700003 ~20ms



Where should the scheduler go?

Disk Scheduling

- Requests to disk are not served in FIFO, they are reordered with other pending requests
- Why? In order to read blocks in sequence as far as possible, to minimize seek time and rotational delay
- Who does scheduling? OS does not know internal geometry of disk, so scheduling done mostly by disk controller

Strategy: always choose request that requires least positioning time (time for seeking and rotating)•Greedy algorithm (just looks for best NEXT decision)

How to implement in disk?

How to implement in OS?

Use Shortest Seek Time First (SSTF) instead

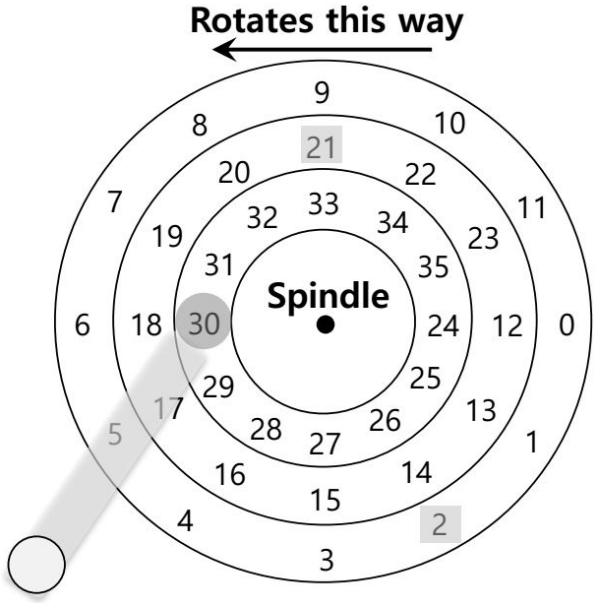
Disadvantages? Easy for far away requests to starve

Shortest Seek Time First (SSTF)

•Access block that we can seek to fastest

–Go to 21 (one track away)before 2 (two tracks away)

•Problem: starvation(some requests that are far from current position of head may never get served)



SSTF: Scheduling Request 21 and 2
Issue the request to 21 --> issue the request to 2

Shortest Seek Time First (SSTF)

Problem 1: The drive geometry is not available to the host OS

Solution: OS can simply implement Nearest-block-first (NBF)

Problem 2: Starvation

If there were a steady stream of request to the inner track, request to other tracks would then be ignored completely.

Elevator/SCAN algorithm

- Disk head does one sweep over tracks and serves requests that fall on the path
- Elevator/SCAN: sweep outer to inner, then inner to outer
- C-SCAN: sweep only one direction (say, outer to inner) and circle back, start again

–Why? Sweeping back and forth favors middle tracks more

- F-SCAN: freeze queue while scanning

–Why? Avoid starving far away requests

Elevator/SCAN algorithm

Elevator Algorithm:

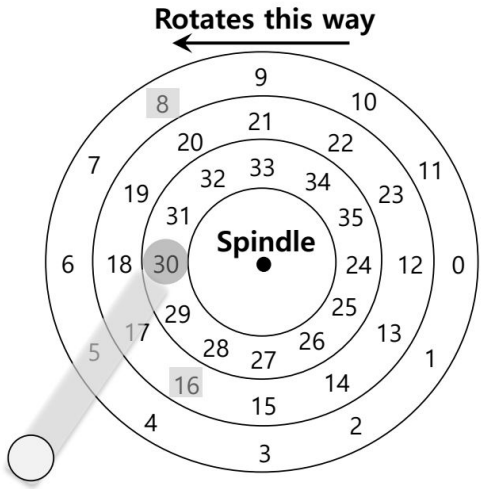
- Sweep back and forth, from one end of disk other, serving requests as pass that cylinder
- Sorts by cylinder number; ignores rotation delays

Pros/Cons?

Better: C-SCAN (circular scan)

- Only sweep in one direction

Shortest Positioning Time First (SPTF)



SSTF: Sometimes Not Good Enough

- If rotation is faster than seek : request 16 → request 8
- If seek is faster than rotation : request 8 → request 16

On modern drives, both seek and rotation are roughly equivalent:
Thus, SPTF (Shortest Positioning Time First) is useful.

- Considers both seek time and rotational latency

–Better to serve 8 before 16,even though seek time is higher

–Why? 16 incurs a much higher rotational latency

Work Conservation

Work conserving schedulers always try to do work if there's work to be done

Sometimes, it's better to wait instead if system anticipates another request will arrive

Such non-work-conserving schedulers are called anticipatory schedulers

Data Storage on Disk

- Bits stored on disk with some error detection/correction bits
  - Correct random bit flips
  - Detect corruption of data
- Disk controller or OS can handle some errors(e.g., blacklisting certain sectors)
- If errors cannot be masked, user perceives hard disk failures
- Technologies such as RAID (Redundant Array of Inexpensive Disks) provide high reliability and performance by replicating across multiple disks.