# Hard Disks

# Basic Interface



Disk has a sector-addressable address space

- Appears as an array of sectors

Sectors are typically 512 bytes or 4096 bytes.

Main operations: reads + writes to sectors

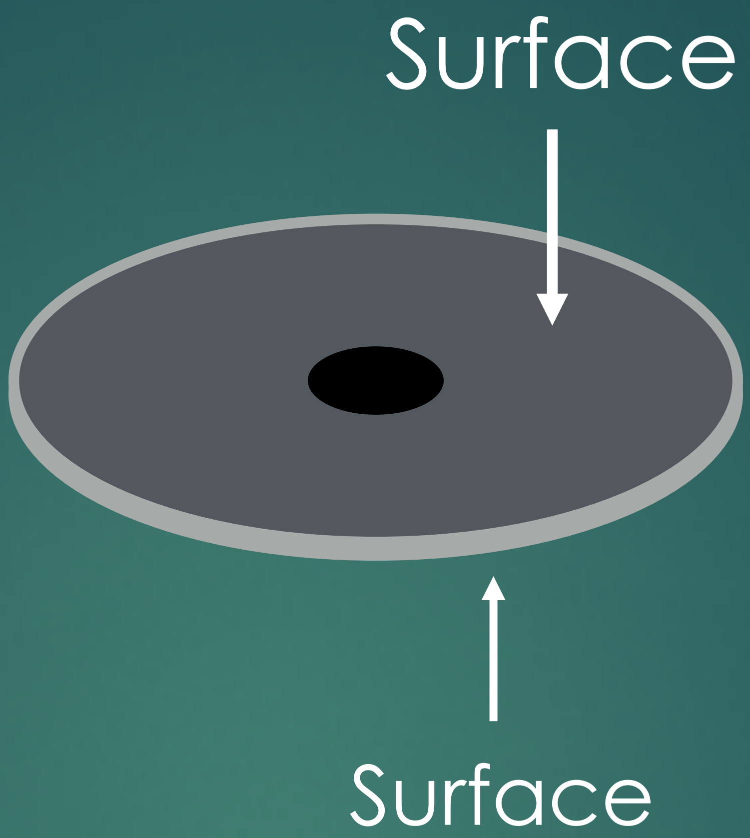
Mechanical (slow) nature makes management "interesting"

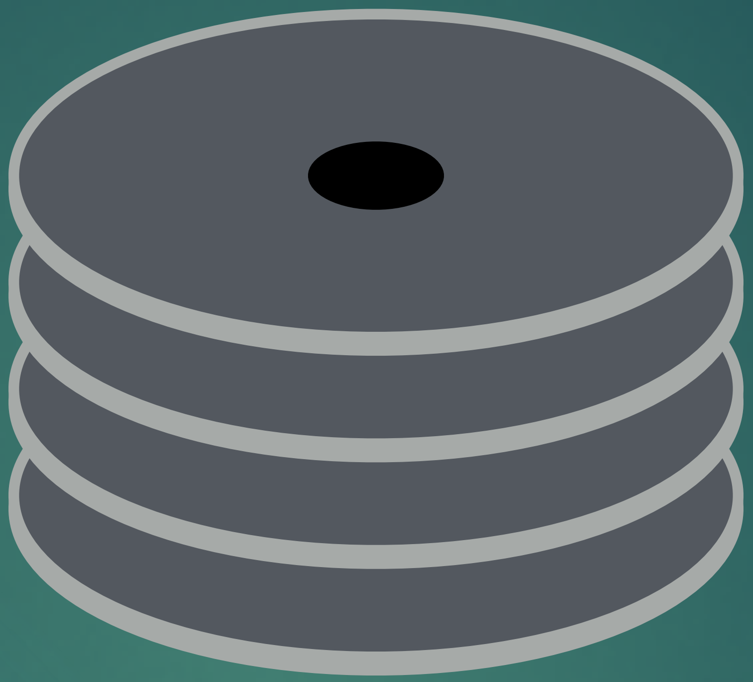
# Disk Internals

Platter

Platter is covered with a magnetic film.

Spindle


Surface
Surface



Many platters may be bound to the spindle.

Each surface is divided into rings called tracks.
A stack of tracks (across platters) is called a cylinder.

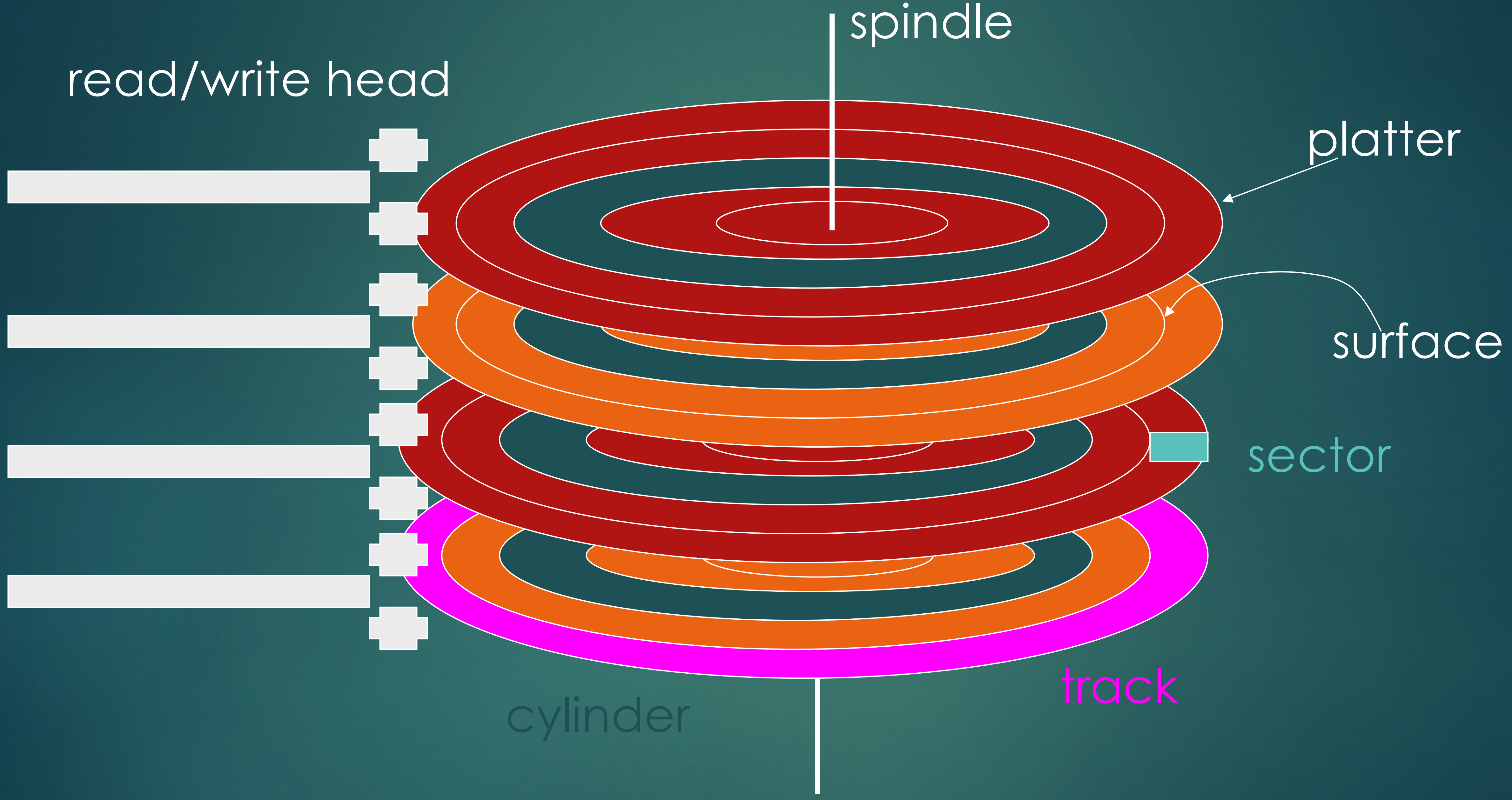

The tracks are divided into numbered sectors.


<u>Heads</u> on a moving <u>arm</u> can read from each surface.


Spindle/platters rapidly spin.

# Disk Terminology





# Hard Drive Demo



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

# Let's Read 12!

# 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 crosses servo burst, figure out location and adjust as needed

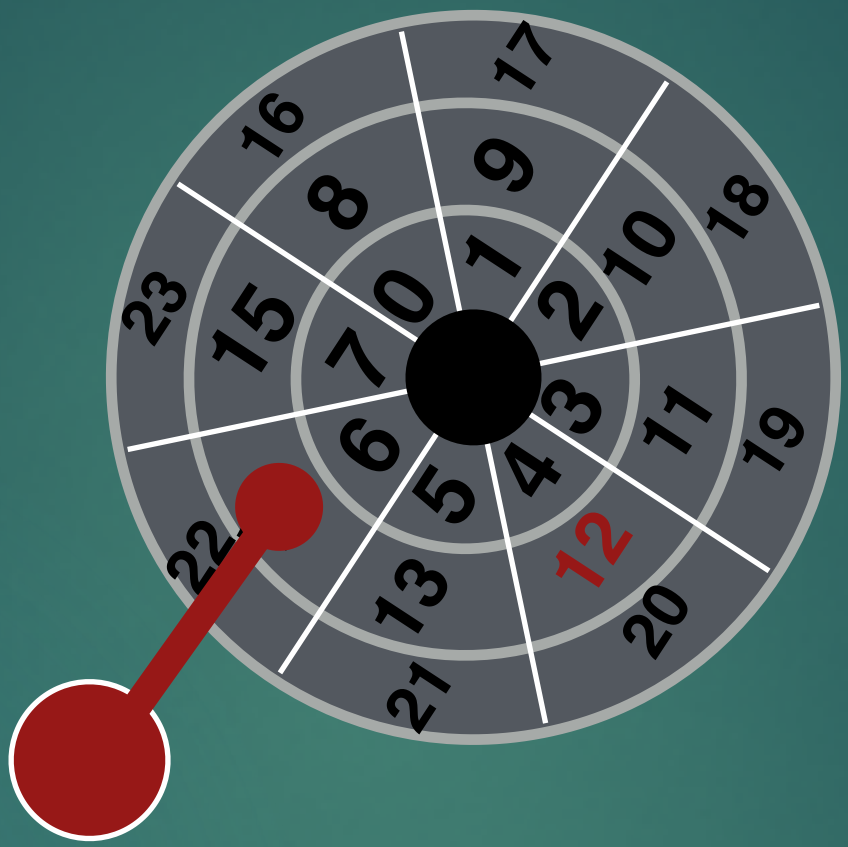
# Let's Read 12!

# Seek to right track.

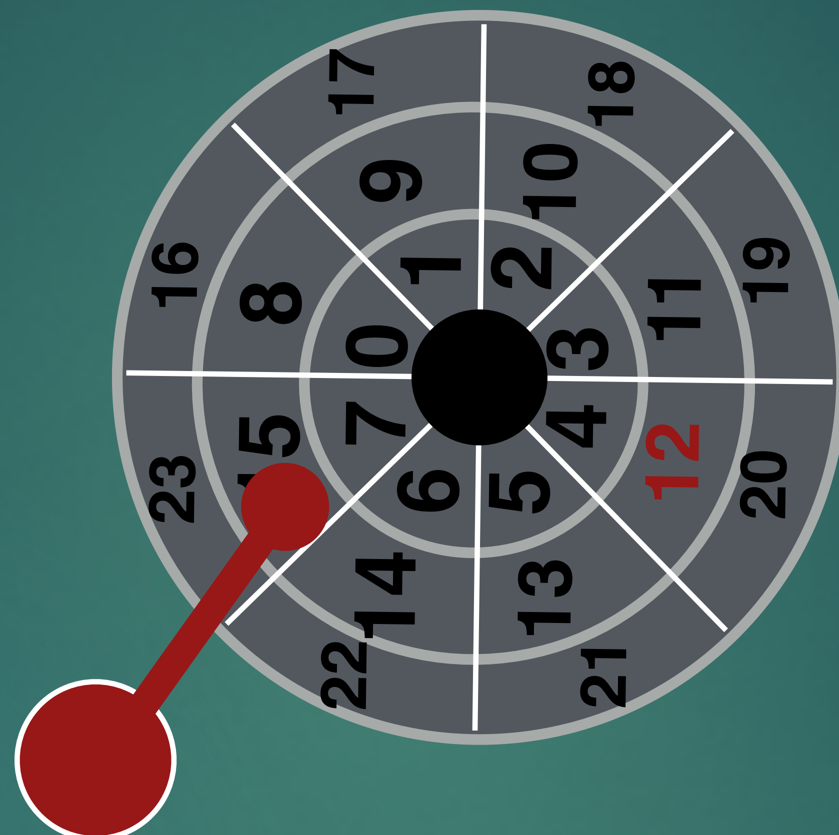
# Seek to right track.

# Seek to right track.

# Wait for rotation.



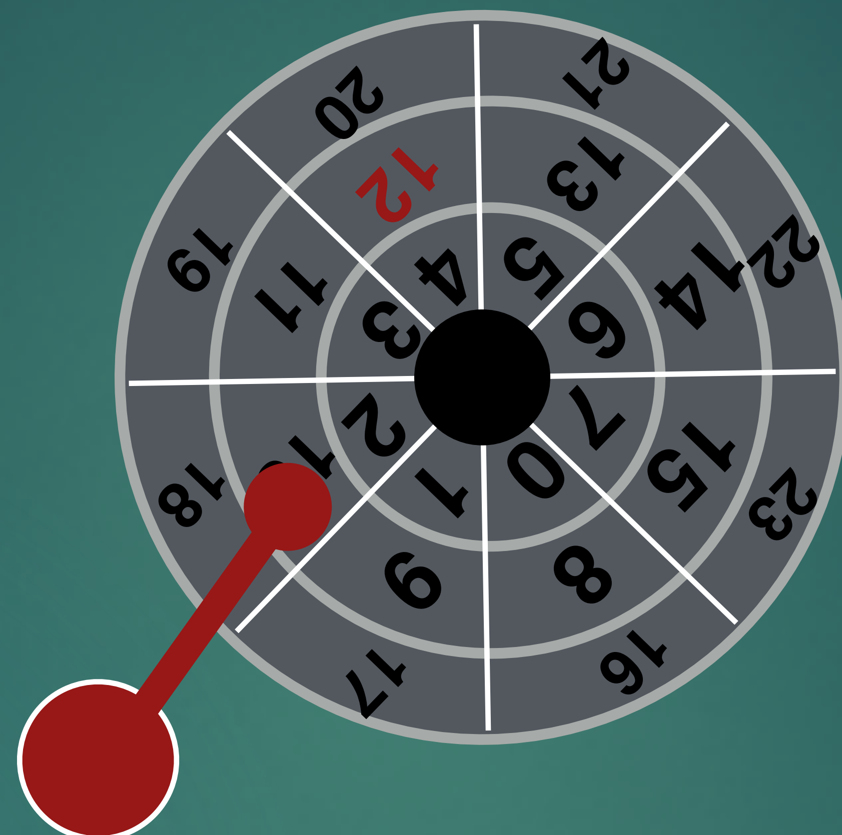
# Wait for rotation.



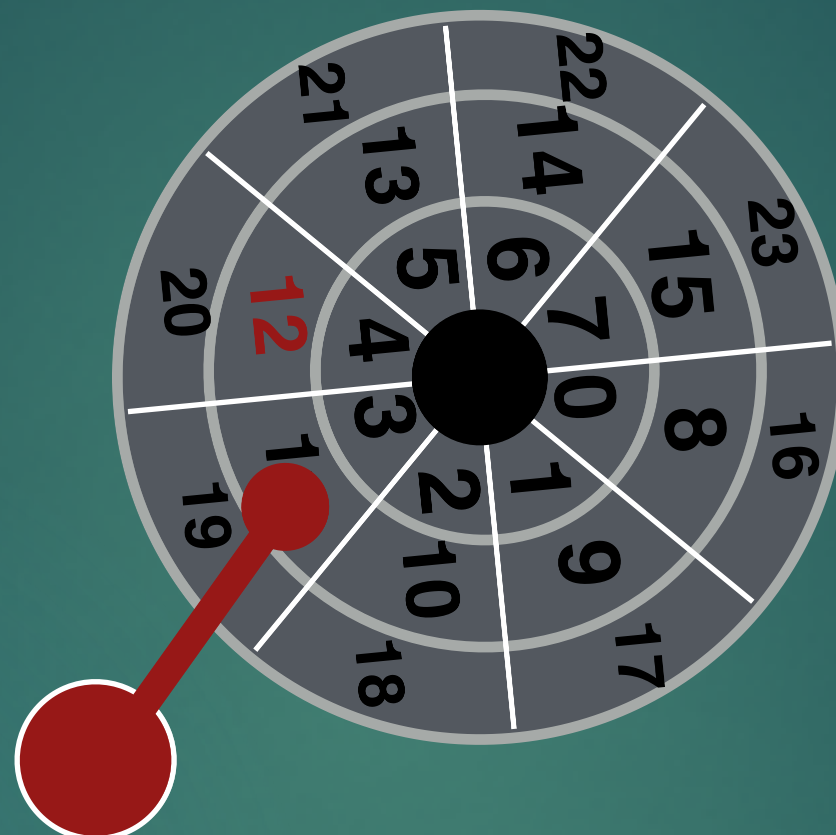
# Wait for rotation.

# Wait for rotation.

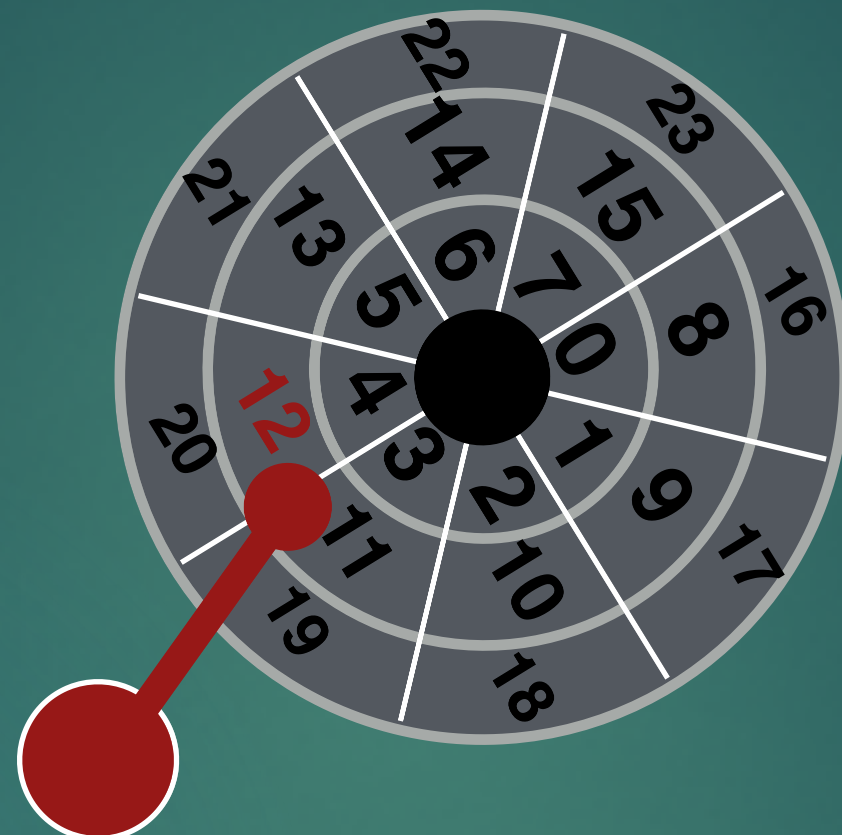
# Wait for rotation.



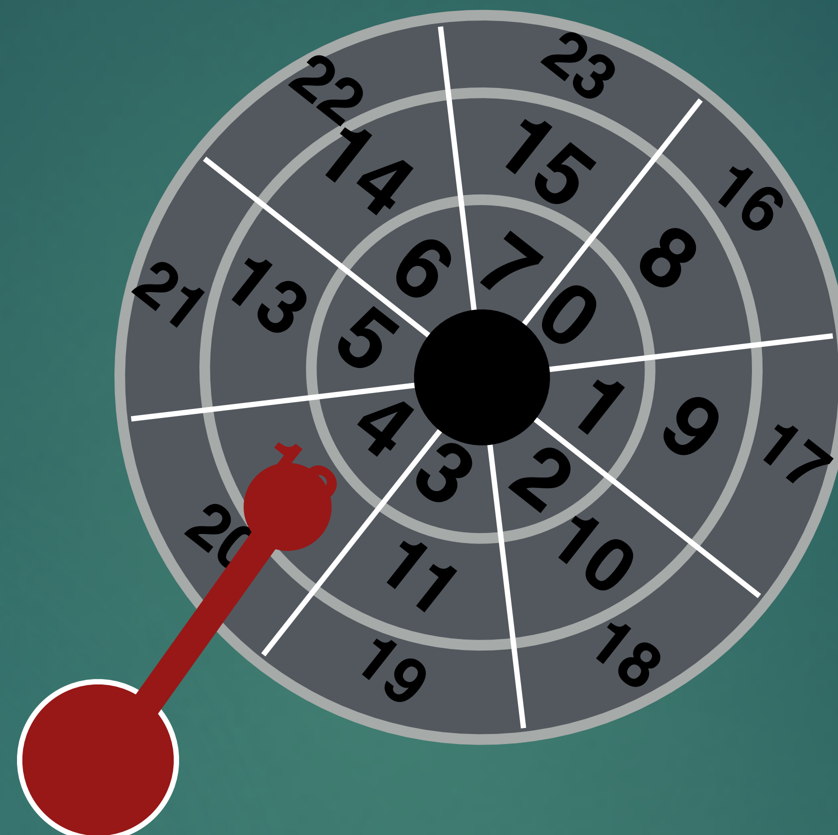
# Wait for rotation.



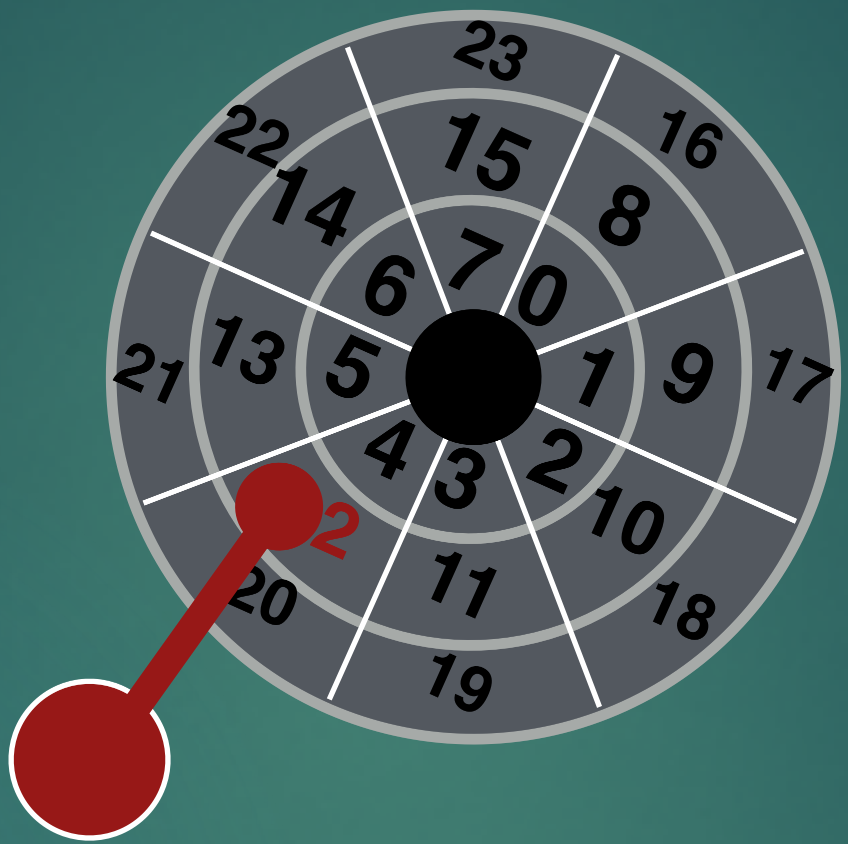
# Transfer data.



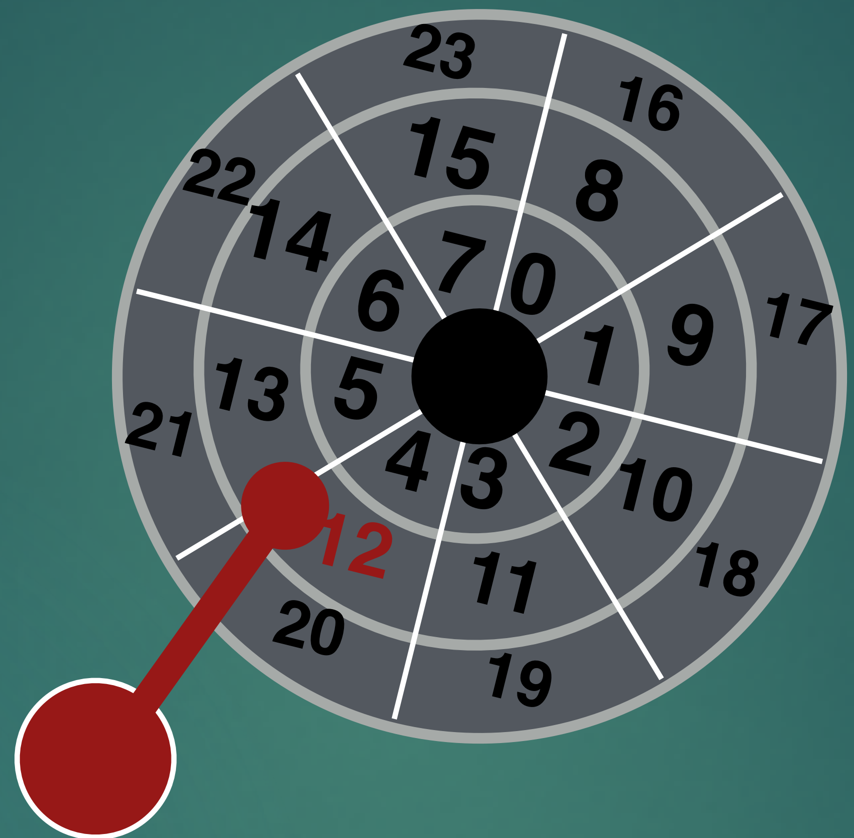
# Transfer data.



# Transfer data.



# Yay!



# Time to Read/write



Three components:

Time = seek + rotation + transfer time

# Seek, Rotate, Transfer



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, Rotate, Transfer



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 =

8.3 ms / rotation

Average rotation?

8.3 ms / 2 = 4.15 ms

# Seek, Rotate, Transfer



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



So…

- 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)

# Disk Spec



| | Cheetah | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 32 MB |

Sequential workload: what is throughput for each?

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

# Disk Spec



| | Cheetah | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 32 MB |

Random workload: what is throughput for each?
(what else do you need to know?)

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

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Seek + rotation + transfer

Seek = 4 ms

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Average rotation in ms?

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

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Transfer of 16 KB?

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

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Cheetah time = 4ms + 2ms + 125us = 6.1ms

Throughput?

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Cheetah time = 4ms + 2ms + 125us = 6.1ms

$$\text{throughput} = \frac{16\ \text{KB}}{6.1\text{ms}} \times \frac{1\ \text{MB}}{1024\ \text{KB}} \times \frac{1000\ \text{ms}}{1\ \text{sec}} = 2.5\ \text{MB/s}$$

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Time = seek + rotation + transfer
Seek = 9ms

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

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

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

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

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Barracuda time = 9ms + 4.1ms + 149us = 13.2ms

| | Cheetah | Barracuda |
|---|---|---|
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |

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

Barracuda time = 9ms + 4.1ms + 149us = 13.2ms

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

| | Cheetah | Barracuda |
|---|---|---|
| Capacity | 300 GB | 1 TB |
| RPM | 15,000 | 7,200 |
| Avg Seek | 4 ms | 9 ms |
| Max Transfer | 125 MB/s | 105 MB/s |
| Platters | 4 | 4 |
| Cache | 16 MB | 32 MB |

| | Cheetah | Barracuda |
|---|---|---|
| Sequential | 125 MB/s | 105 MB/s |
| Random | 2.5 MB/s | 1.2 MB/s |

# Other Improvements

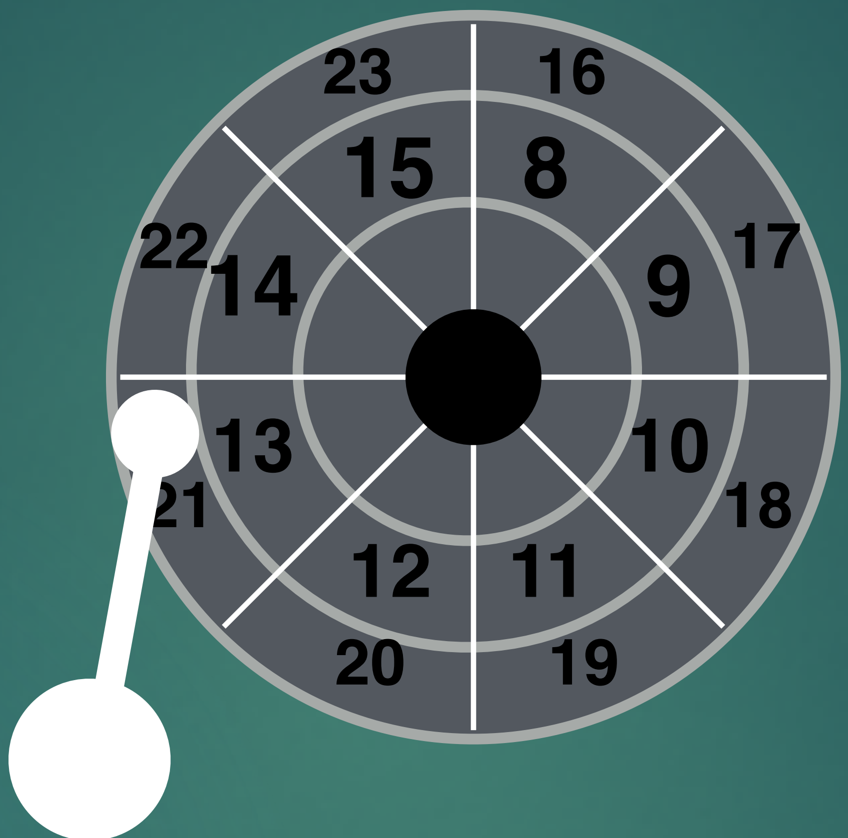


Track Skew

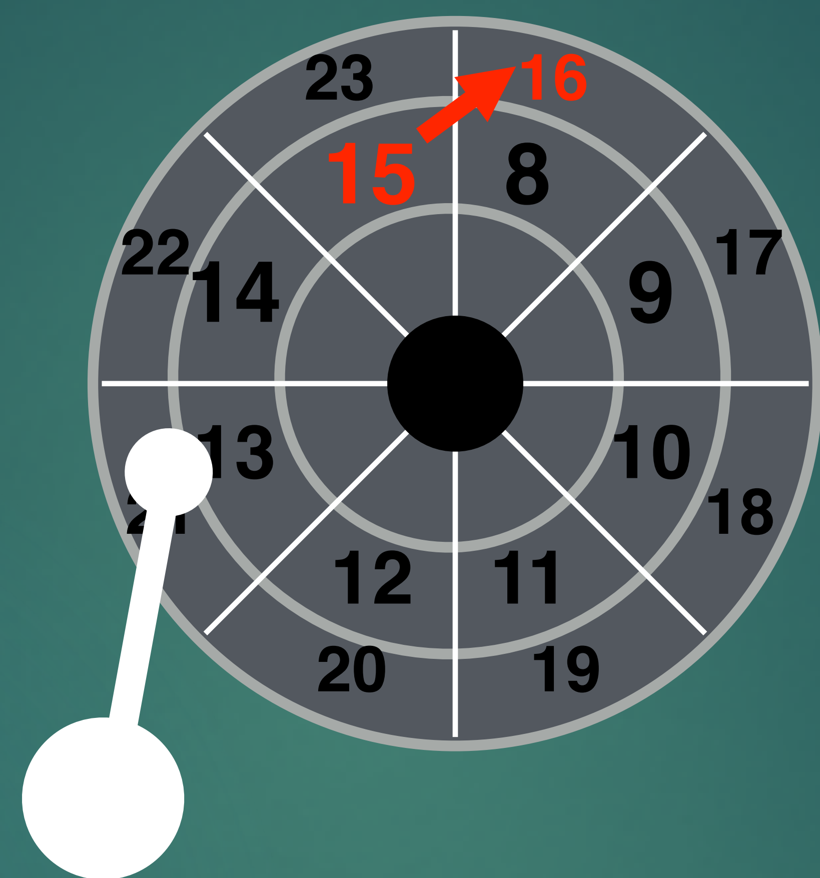
Zones

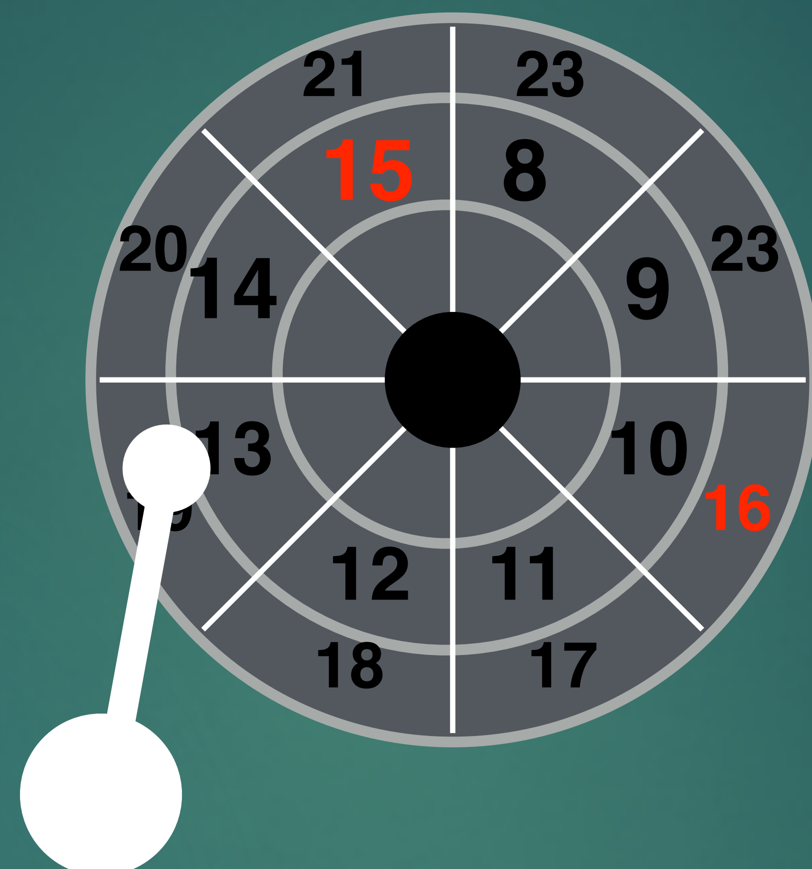
Cache

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



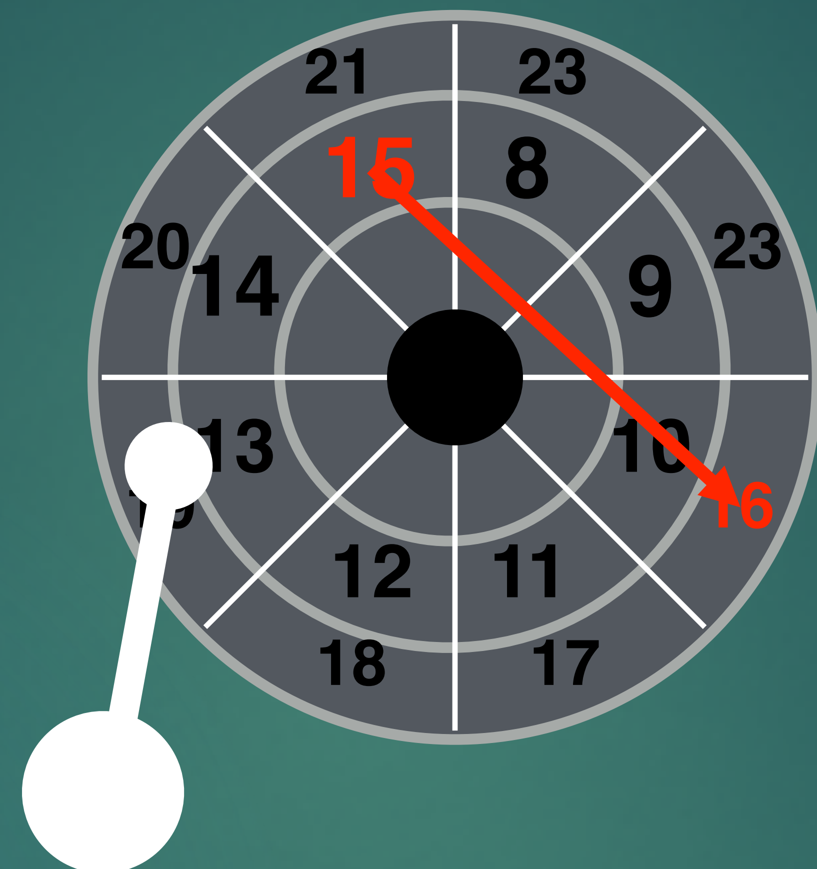
When reading 16 after 15, the head won't settle quick enough, so we need to do a rotation.





21
23
15
8
20
14
9
23
13
10
16
12
11
18
17

# enough time to settle now



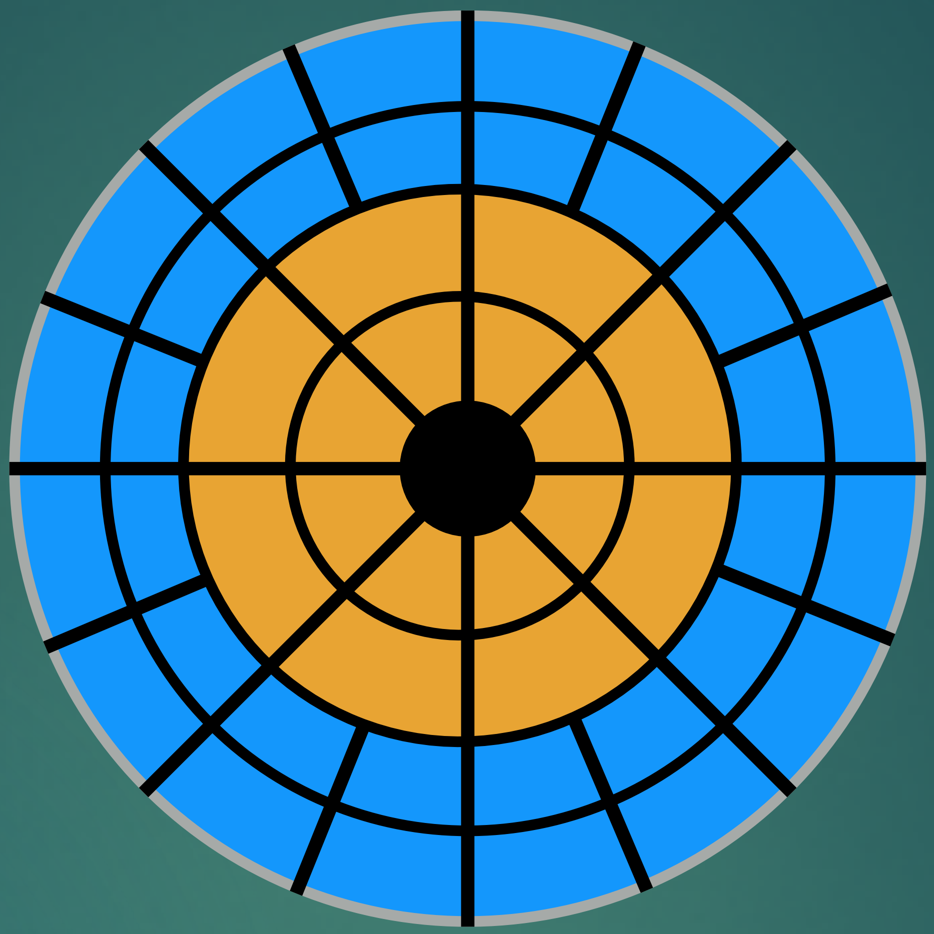
# Other Improvements



Track Skew

Zones

Cache

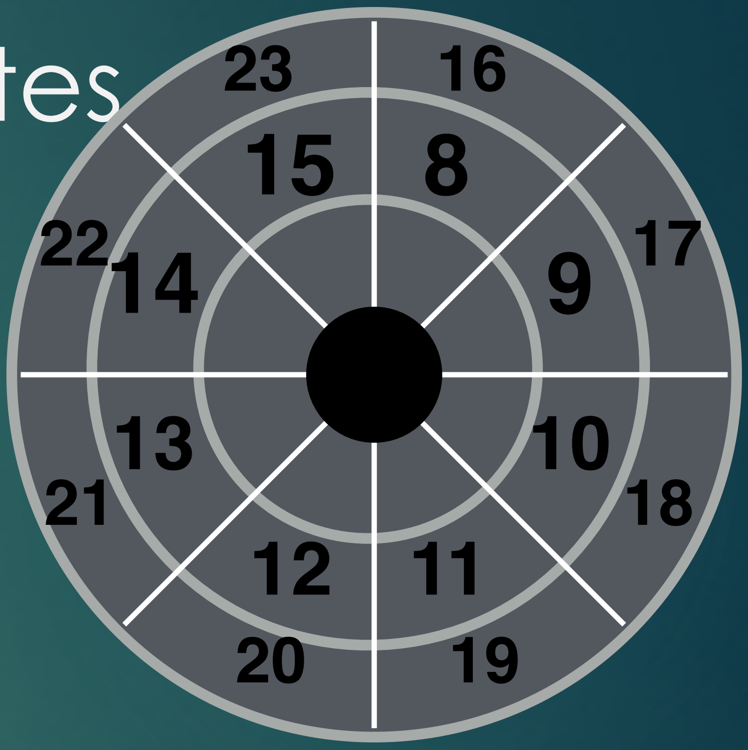


ZBR (Zoned bit recording): *More sectors on outer tracks*

# 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: 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

# I/O Schedulers

# I/O Schedulers



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

# FCFS (First-Come-First-Serve)

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

# FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms for random request

How long (roughly) does the below workload take?

- Requests are given in sector numbers

~60ms

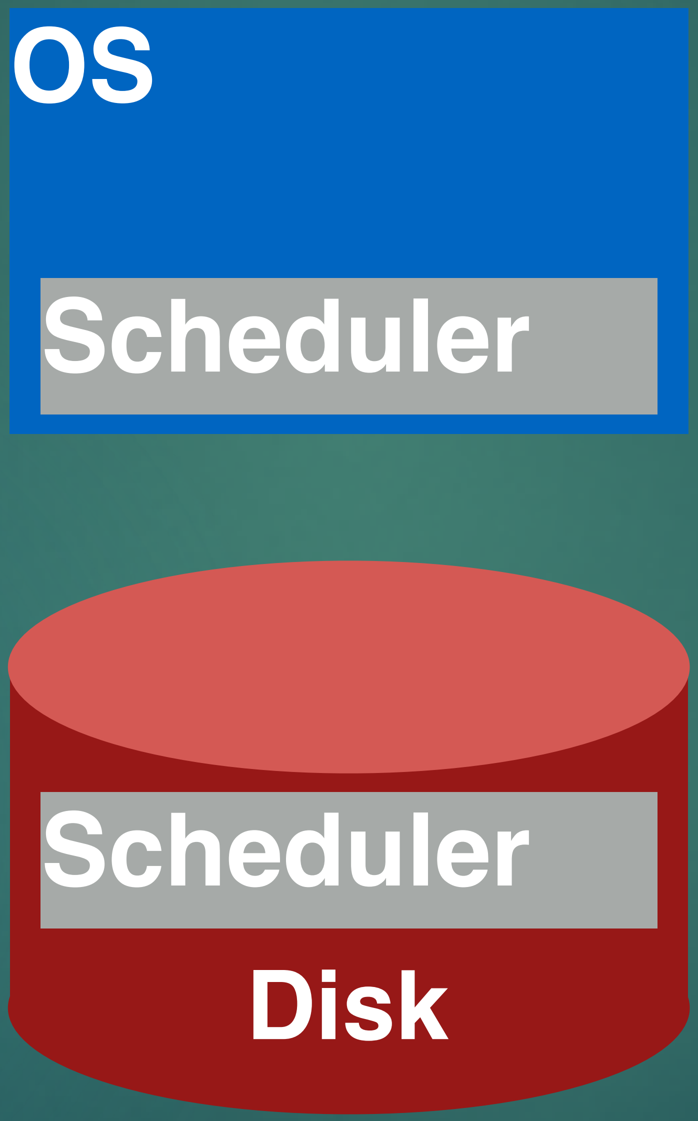
300001, 700001, 300002, 700002, 300003, 700003

300001, 300002, 300003, 700001, 700002, 700003

~20ms

# Schedulers






Where should the scheduler go?

# SPTF (Shortest Positioning Time First)



**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**

# SCAN



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

# What happens?



Assume 2 processes each calling read() with C-SCAN

```
void reader(int fd) {
    char buf[1024];
    int rv;
    while((rv = read(buf)) != 0) {
        assert(rv);
        // takes short time, e.g., 1ms
        process(buf, rv);
    }
}
```

# 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

# CFQ (Linux Default)



Completely Fair Queueing

- Queue for each process
- Weighted round-robin between queues, with slice time proportional to priority
- Yield slice only if idle for a given time (anticipation)

Optimize order within queue

# I/O Device Summary



Overlap I/O and CPU whenever possible!

- use interrupts, DMA

Storage devices provide common block interface

On a disk: Never do random I/O unless you must!

- e.g., Quicksort is a terrible algorithm on disk

Spend time to schedule on slow, stateful devices