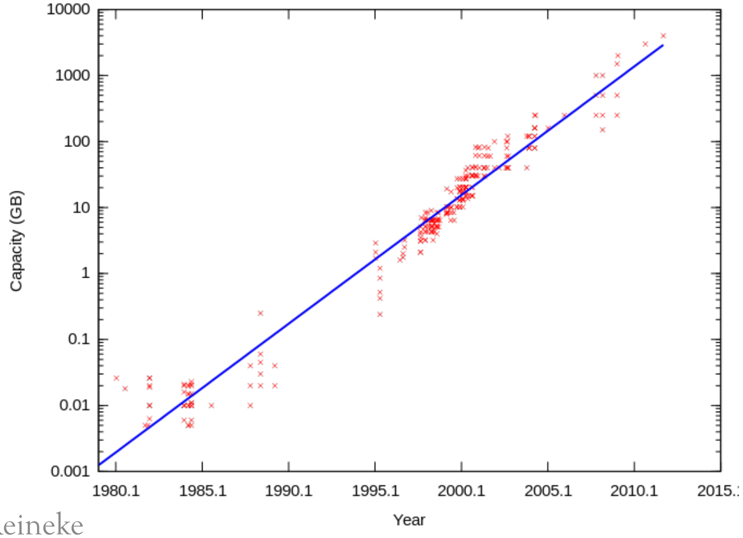
Persistence:
Disks + I/O Scheduling

OSTEP Chapter 37:
http://pages.cs.wisc.edu/~remzi/OSTEP/file-disks.pdf

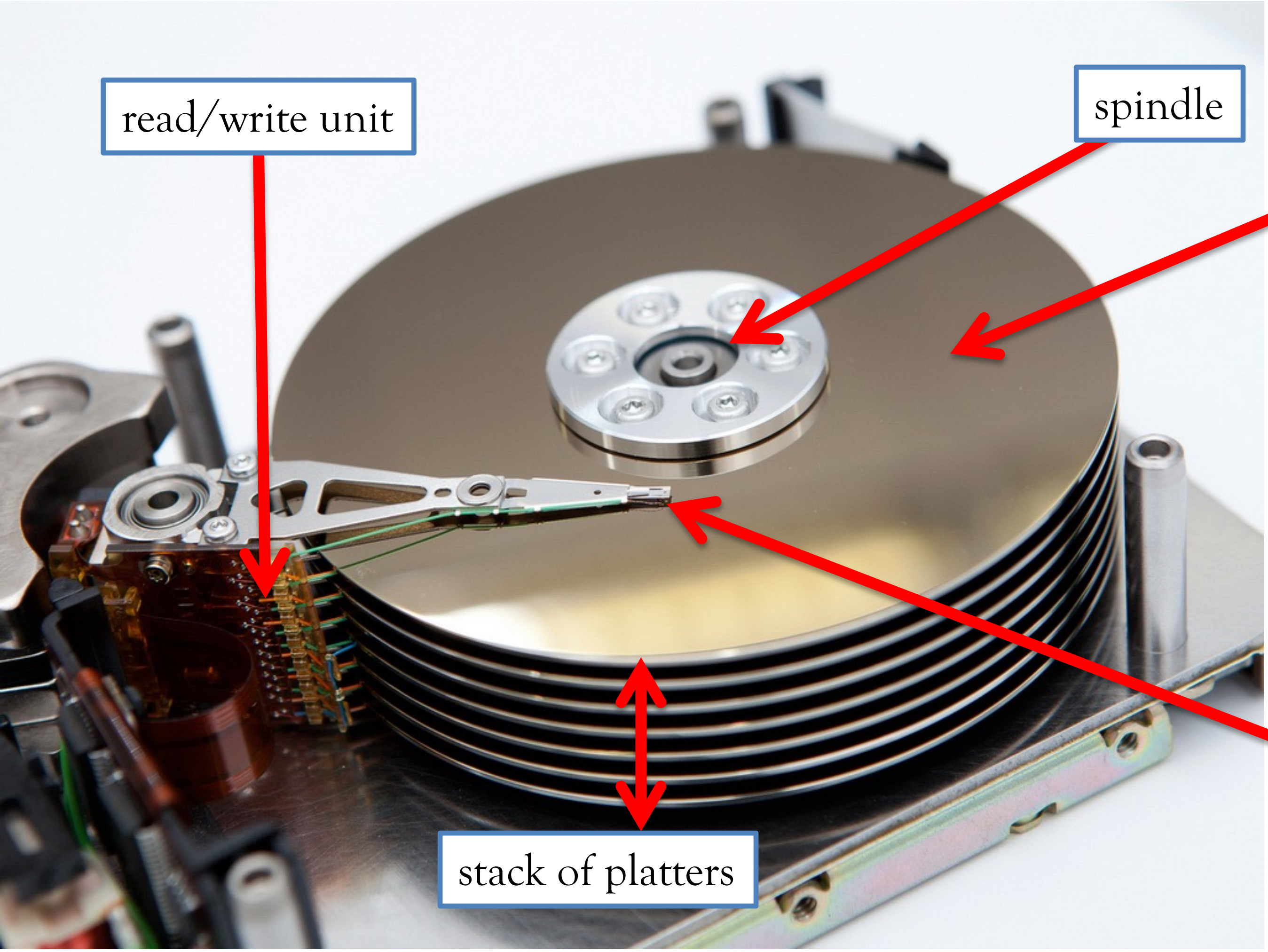
Jan Reineke
Universität des Saarlandes

Hard disk: Properties

- **Persistent** memory:
  Data persists without power supply
  (in contrast to DRAM and SRAM)
- **Very high capacities:**
  Doubling about every 2 years:
- **Much slower** than DRAM



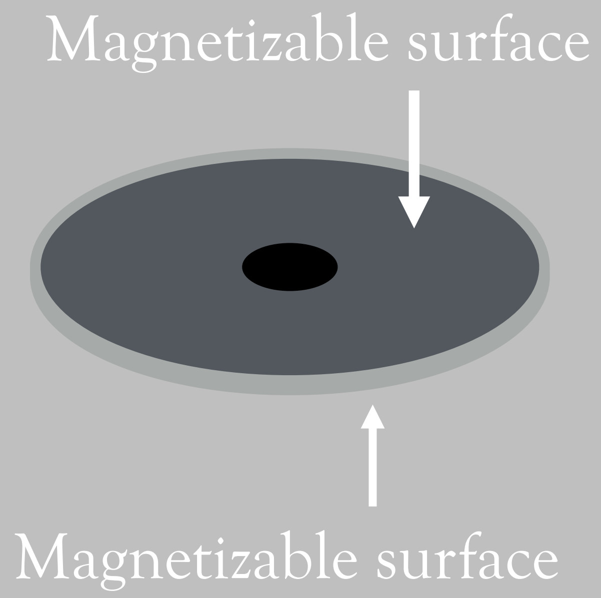
Hard disk: Physical structure



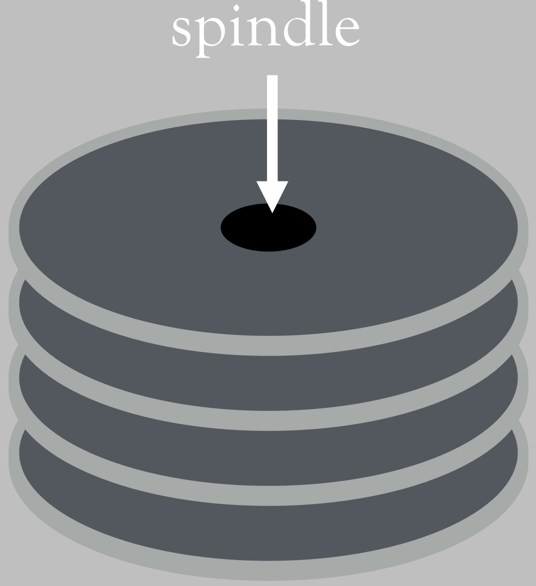
- magnetic coating on both sides
- reading and writing is contactless
- fast rotation:
  up to 15000 revolutions per minute (RPM)

- moved by rotating read/write unit

Hard disk geometry



Hard disk geometry

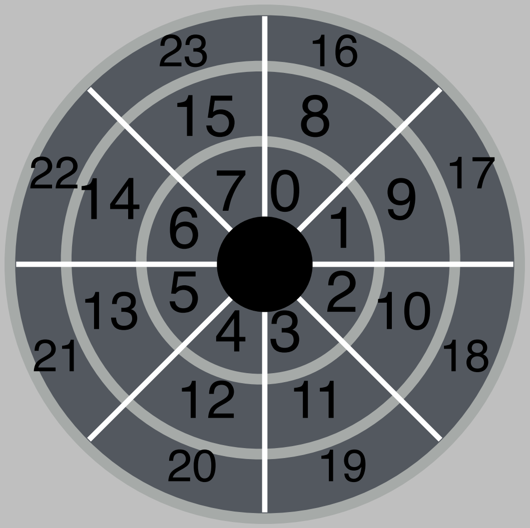


Stack of platters connected to each other via spindle.

Hard disk geometry

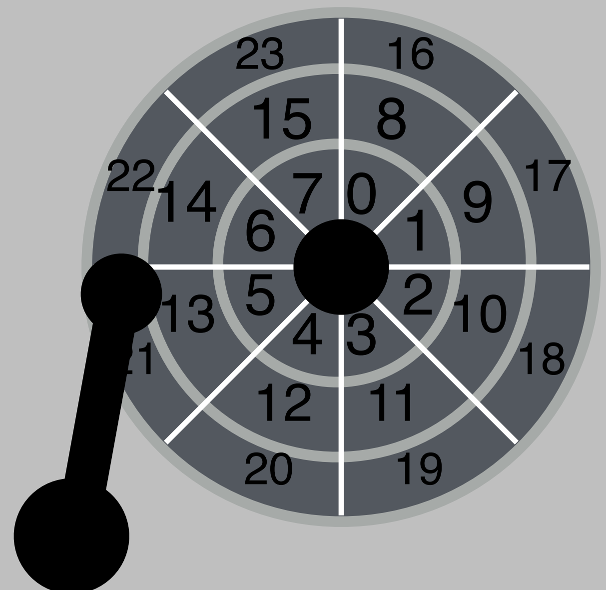
Each surface is divided into rings called <u>tracks</u>.
A stack of tracks (across platters) is called a <u>cylinder</u>.

Hard disk geometry



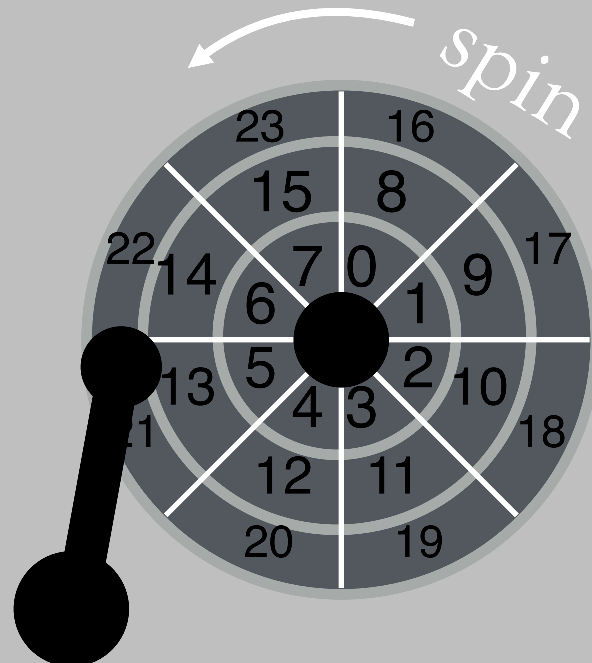
The cylinders are divided into numbered <u>sectors</u>.

Hard disk geometry



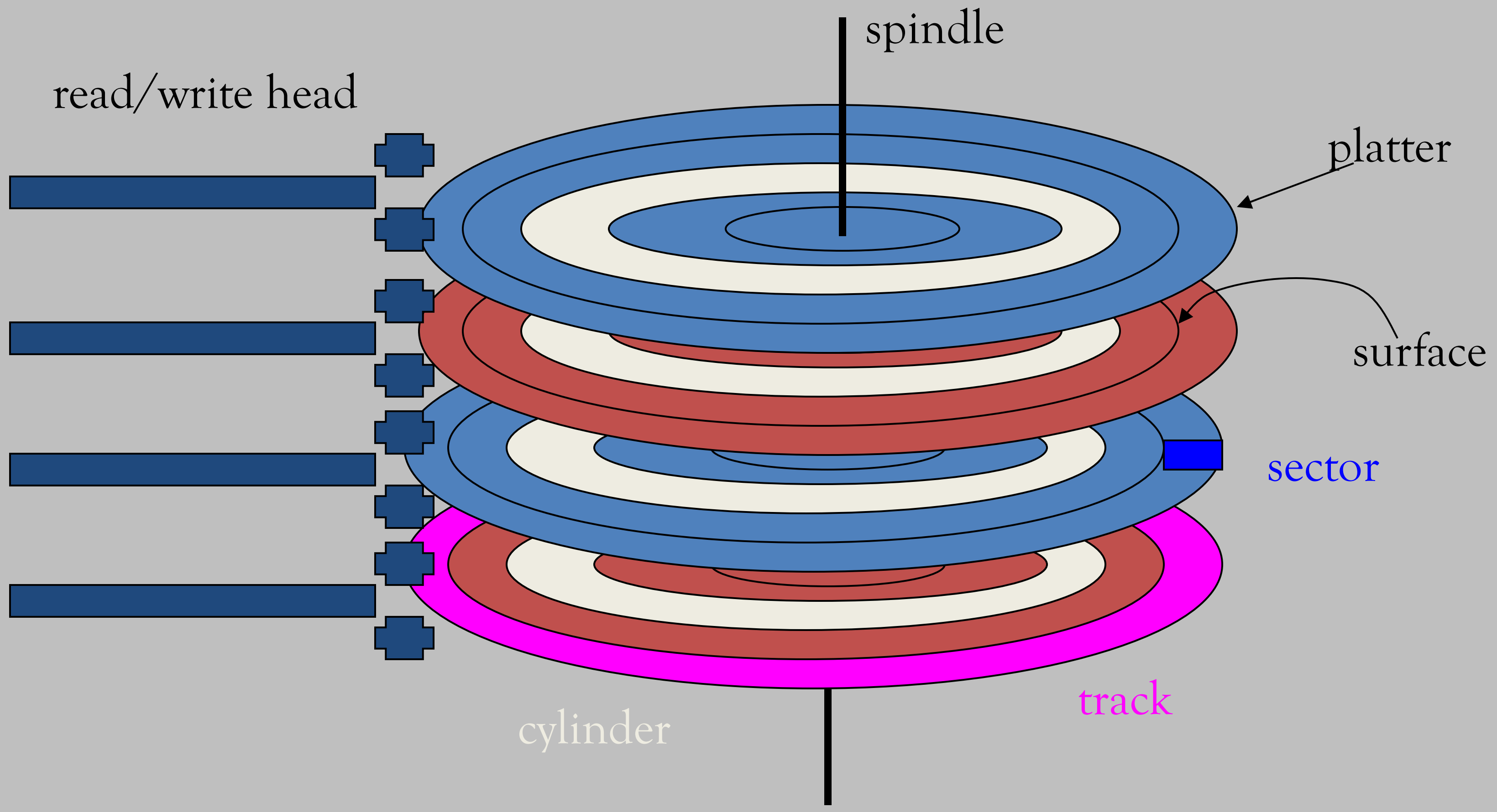
<u>(Read/write) heads</u> on a moving <u>arm</u> can read from each surface.

Hard disk geometry



Spindle/platters rapidly spin.

Disk terminology



Basic interface

Disk has a sector-addressable address space

– Appears as an **array of sectors**

Sectors are typically <u>512 bytes</u> or 4096 bytes

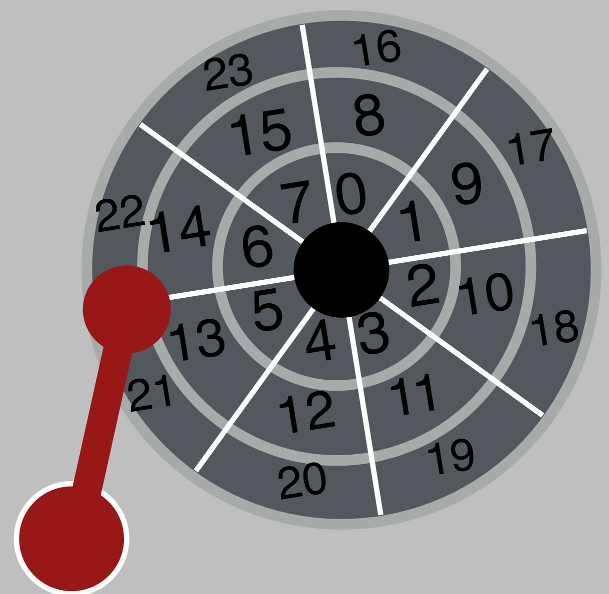
→ thus a page size of 4 KB is sensible

Main operations:

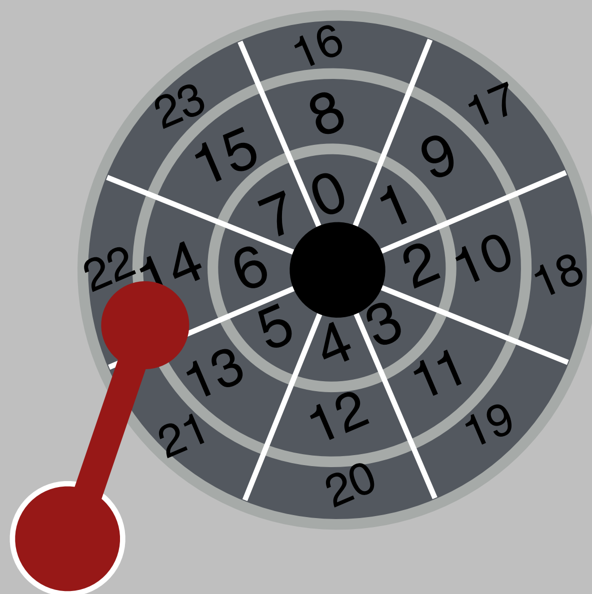
Reads and **writes** to sectors

Let's read sector 12

Seek to right track

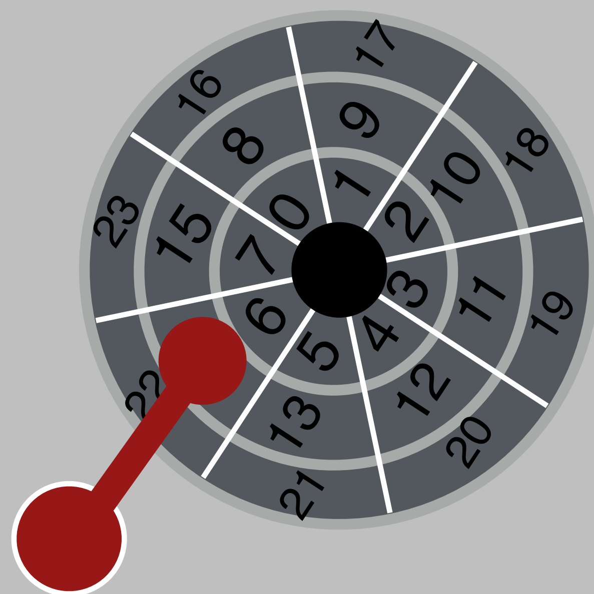


Seek to right track

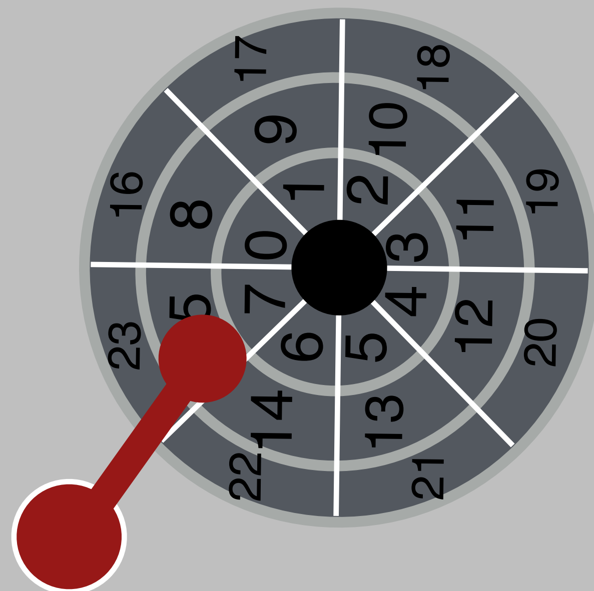


Seek to right track

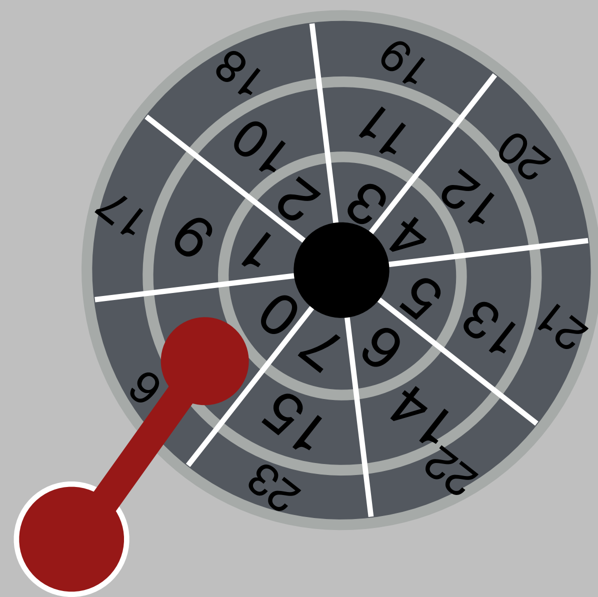
Wait for rotation



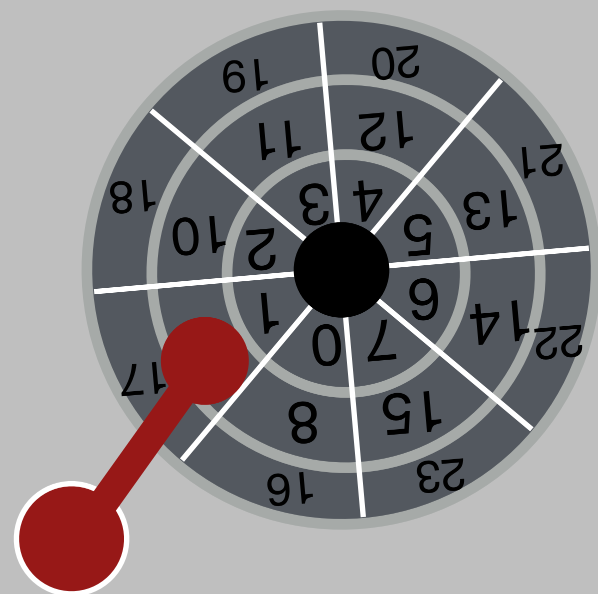
Wait for rotation



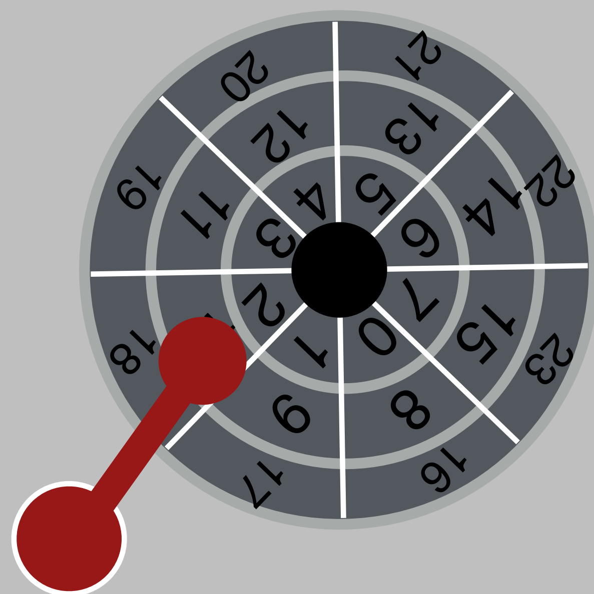
Wait for rotation



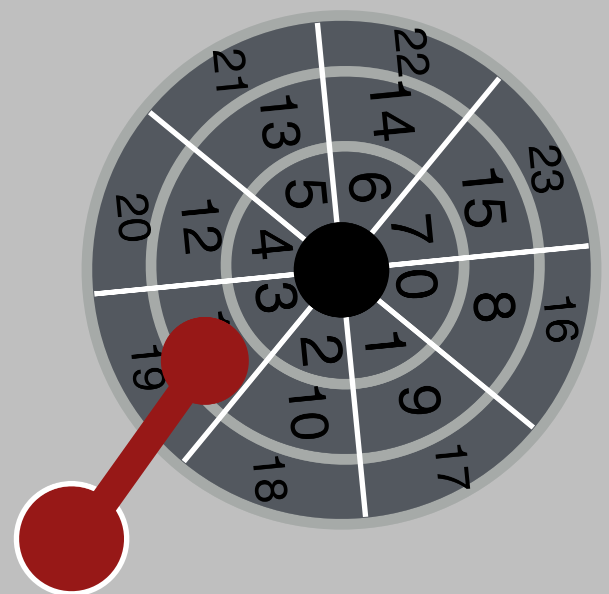
Wait for rotation



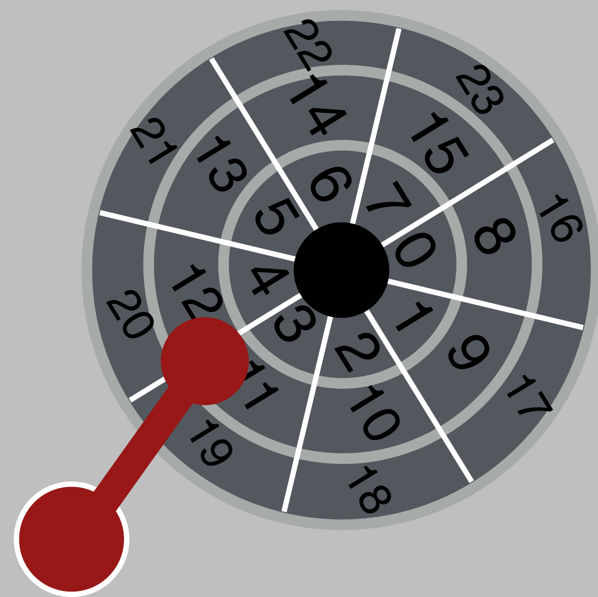
Wait for rotation



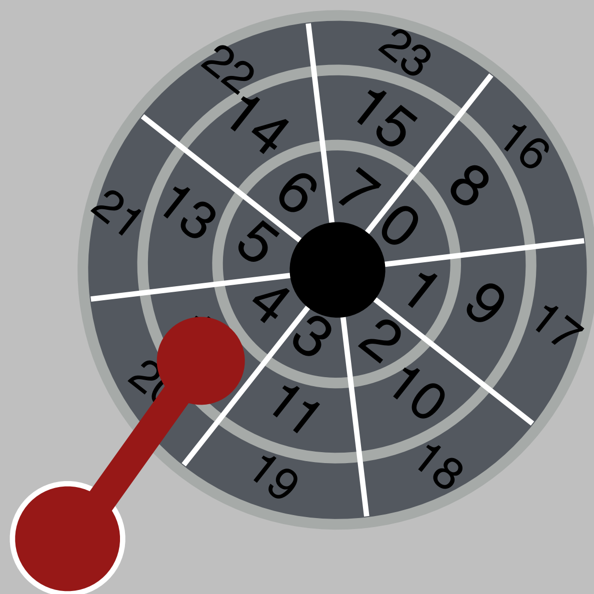
Wait for rotation



Data transfer

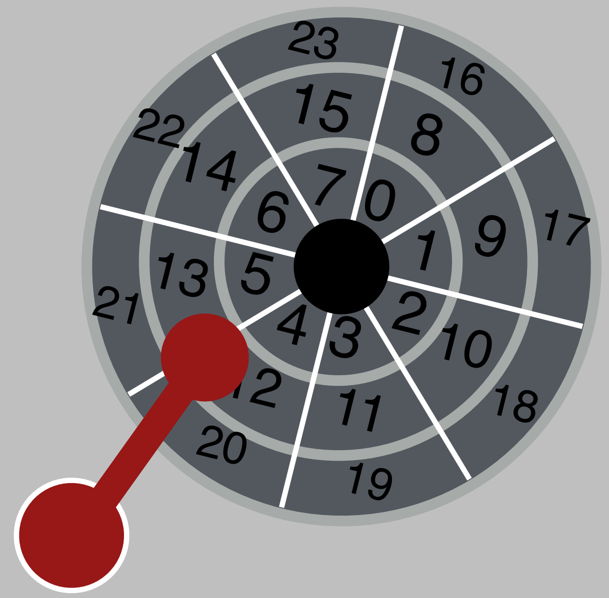


Data transfer



Data transfer

Yay!



Time to read/write

Three components:

- Seek
- Rotation
- Transfer time

Seek, Rotate, Transfer

Depends on **cylinder distance**:

Seek time: between 4 and 10 ms

Seek, **Rotate**, Transfer

Depends on revolutions per minute (RPM)

- 7200 RPM (revolutions per minute) are typical
- 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 time?

8.3ms / 2 = 4.15 ms

Seek, Rotate, **Transfer**

Is transfer time faster or slower than rotate time?

Seek, Rotate, **Transfer**

Pretty fast – depends on RPM and sector density

Maximal transfer rate often 500+ MB/s

How long to transfer 512 bytes?

512 bytes / (500 MB/s) ≈ 1 microsecond (10^{-6} second)

Performance depends on workload

So:

- Seeks are slow
- Rotations are slow
- Transfers are fast → explains large access granularity

What kind of workload is fastest for disk?

- *Sequential*: access sectors in order (transfer dominated)
- *Random*: access sectors arbitrarily
  (seek+rotation dominate)

Disk specifications

	Toshiba AL14SXB (2017)	Seagate Exos X14 (2018)
Capacity	900 GB	14 TB
RPM	15.000	7.200
Average seek time	2.0 ms	4.16 ms
Max. transfer rate	290 MB/s	261 MB/s
Platters	?	8
Cache	128 MB	256 MB

Sequential workload: what is throughput for each?

Toshiba: 290 MB/s.

Seagate: 261 MB/s.

Disk performance

	Toshiba AL14SXB (2017)	Seagate Exos X14 (2018)	IBM PC/AT (1986)
Capacity	900 GB	14 TB	30 MB
RPM	15.000	7.200	?
Average seek time	2.0 ms	4.16 ms	30-40 ms
Max. transfer rate	290 MB/s	261 MB/s	0.7-1 MB/s (estimated)
Platters	?	8	?
Cache	128 MB	256 MB	?

Throughput on **random** workload?
(What else do you need to know?)

What is size of each random read?
Assume: 4 KB reads

Throughput on random workload

	Toshiba AL14SXB	Seagate Exos X14
RPM	15.000	7.200
Average seek time	2.0 ms	4.16 ms
Max. transfer rate	290 MB/s	261 MB/s

How long does an average random 4 KB read take w/ Toshiba?

Access latency = Seek + rotation + transfer

Seek = 2 ms

Throughput on random workload

	Toshiba AL14SXB	Seagate Exos X14
RPM	15.000	7.200
Average seek time	2.0 ms	4.16 ms
Max. transfer rate	290 MB/s	261 MB/s

$$\text{Rotation} = \frac{1}{2} \times \frac{1\ \text{min}}{15000} \times \frac{60\ \text{s}}{1\ \text{min}} \times \frac{1000\ \text{ms}}{1\ \text{s}} = 2\ \text{ms}$$

Throughput on random workload

	Toshiba AL14SXB	Seagate Exos X14
RPM	15.000	7.200
Average seek time	2.0 ms	4.16 ms
Max. transfer rate	290 MB/s	261 MB/s

$$\text{Transfer} = \frac{1\ \text{s}}{290\ \text{MB}} \times 4\ \text{KB} \times \frac{1.000.000\ \text{us}}{1\ \text{s}} \approx 14\ \text{us}$$

Throughput on random workload

How long does an average random 4 KB read take w/ Toshiba?

Access latency = Seek + rotation + transfer

= 2ms + 2ms + 0.014 ms ≈ 4.0 ms

Throughput?

$$\text{Throughput} = \frac{4\text{ KB}}{4{,}0\text{ ms}} \times \frac{1\text{ MB}}{1024\text{ KB}} \times \frac{1000\text{ ms}}{1\text{ s}} = 1\text{ MB/s}$$

Throughput on random workload: Seagate

How long does an average random 4 KB read take w/ Seagate?

Access latency = Seek + rotation + transfer

= 4.16ms + 4.16ms + 0.015 ms = 8.5 ms

Throughput?

$$\text{Throughput} = \frac{4\ \text{KB}}{8.5\ \text{ms}} \times \frac{1\ \text{MB}}{1024\ \text{KB}} \times \frac{1000\ \text{ms}}{1\ \text{s}} = 0.47\ \text{MB/s}$$

Throughput for different workloads

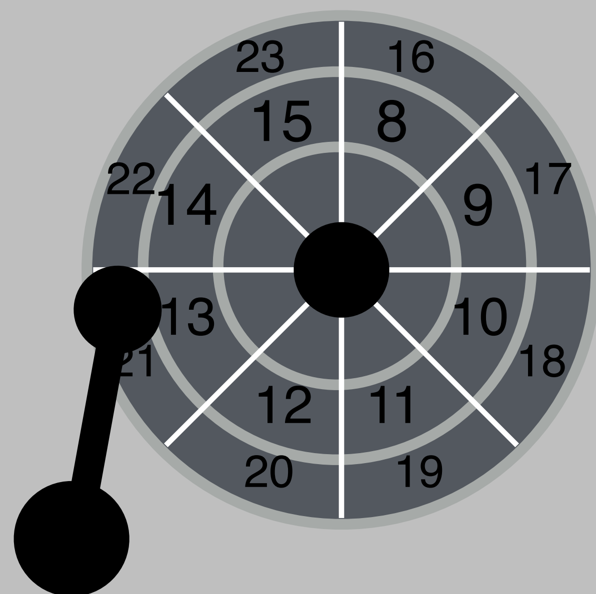
	Toshiba AL14SXB (2017)	Seagate Exos X14 (2018)
Capacity	900 GB	14 TB
RPM	15.000	7.200
Average seek time	2.0 ms	4.16 ms
Max. transfer rate	290 MB/s	261 MB/s
Platters	?	8
Cache	128 MB	256 MB

Workload	Toshiba AL14SXB	Seagate Exos X14
Sequential	290 MB/s	261 MB/s
Random	1 MB/s	0,47 MB/s

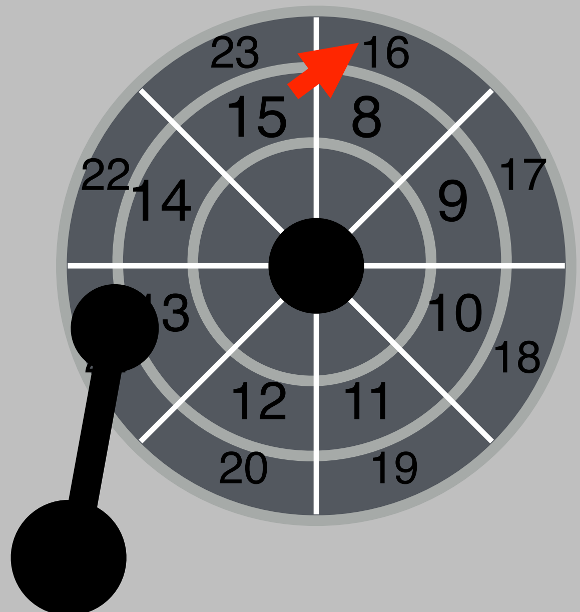
Other improvements

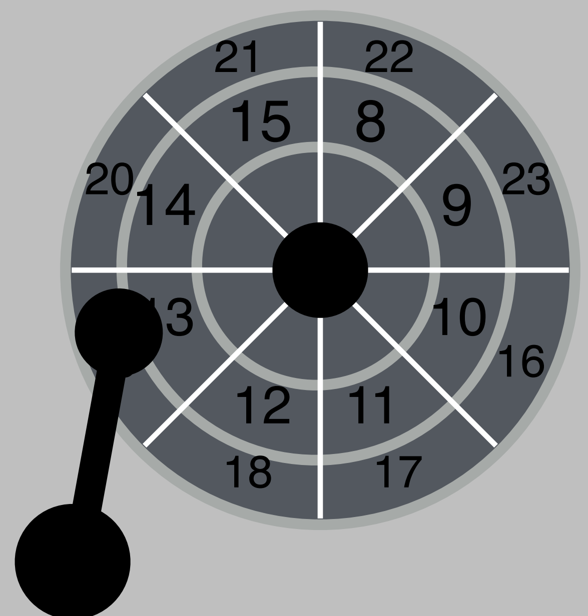
- **Track skew**
- Zones
- Cache

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

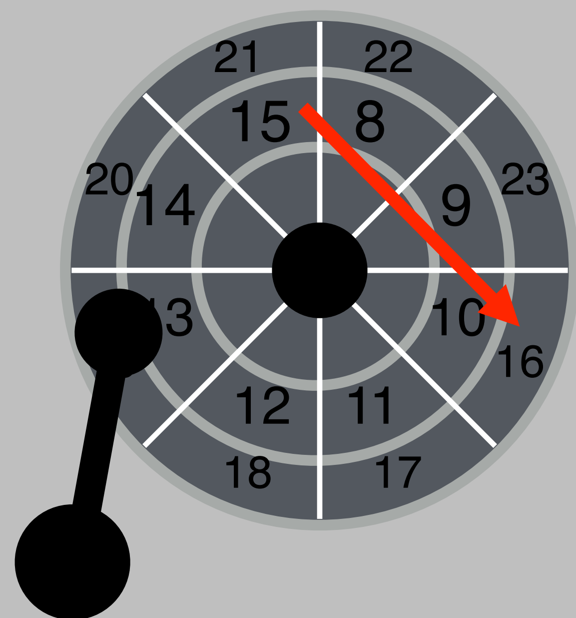


When reading 16 after 15, the head won't settle quick enough, so we need to do a rotation.



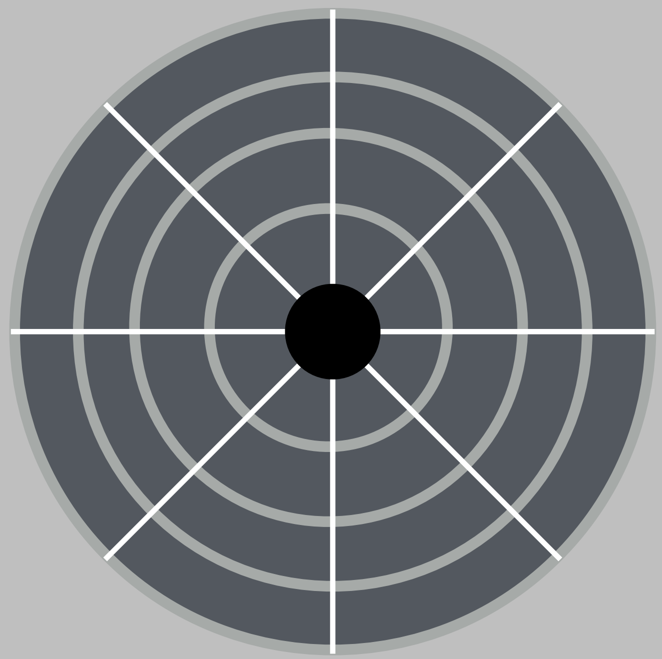

21
22
15
8
20
14
9
23
13
10
16
12
11
18
17

Enough time to settle now!

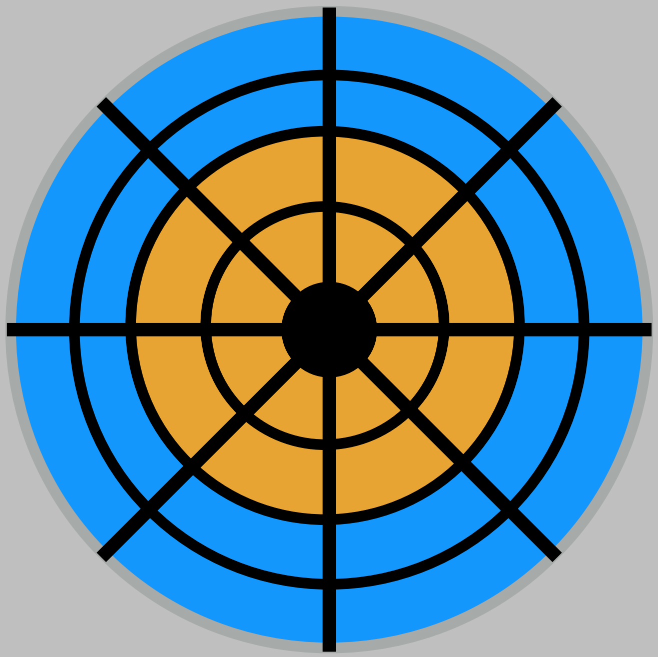


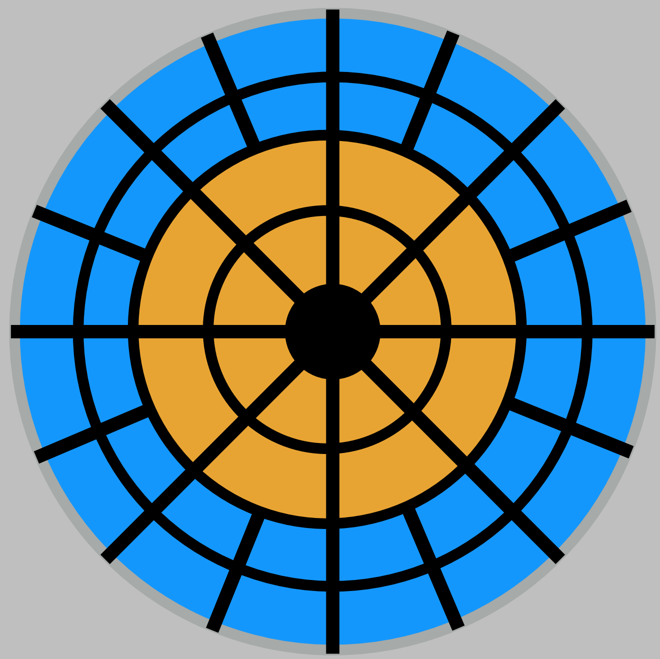
Other improvements

- Track skew
- **Zones**
- Cache



Observation?





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

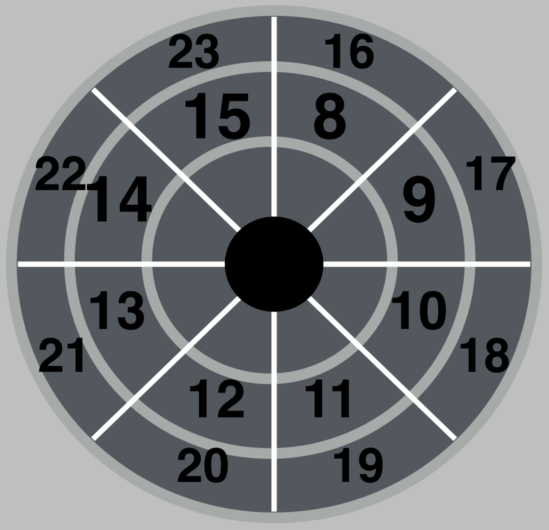
Other improvements

- Track skew
- Zones
- **Cache**

Drive cache

Drives may cache both reads and writes.

OS caches data, too.



What **advantage/disadvantage** does caching in drive have over caching on CPU?

Disk cache

Disk contains internal memory (2-256 MB) used as cache

Read ahead: "track buffer"

Read contents on entire track into memory during rotational delay

Reordering requests:

- Accept new requests before having finished previous ones
- Disk can reorder (schedule) requests for better performance

I/O Schedulers

In what order should I/O requests be served?

Difference to CPU scheduling?

Position of read/write head matters more than length of job

First Come, First Serve (FCFS)

Assume:
Seek + rotation = 10 ms for random request

How long does the below workload take?

Requests are given in sector numbers:

300001, 700001, 300002, 700002, 300003, 700003

≈ 60ms

First Come, First Serve

Assume:
Seek + rotation = 10 ms for random request

How long does the below workload take?

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

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

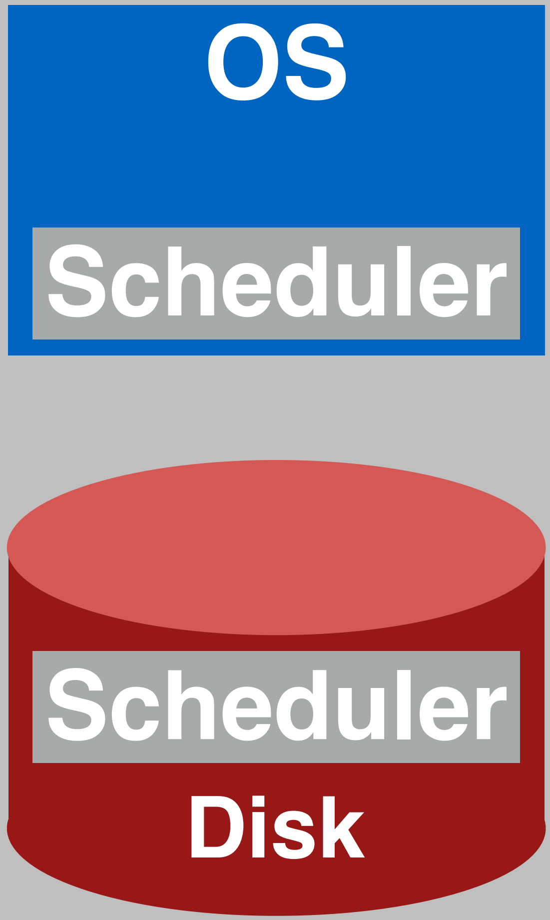
Shortest Job First?

In contrast to CPU scheduling we know in advance how long an access will take!

Analog to SJF: **Shortest Positioning Time First**

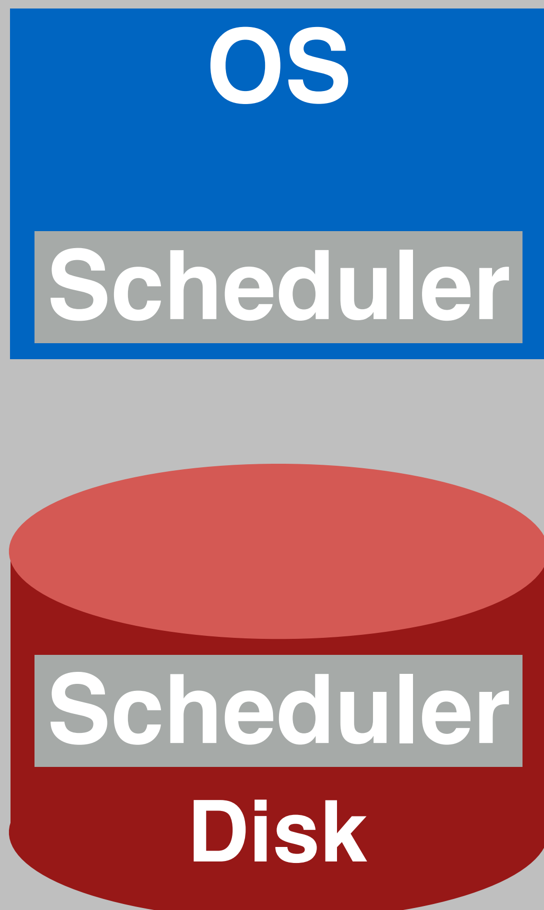
Here: Positioning = Seek + Rotation

How to implement SPTF?



Where should scheduler go?

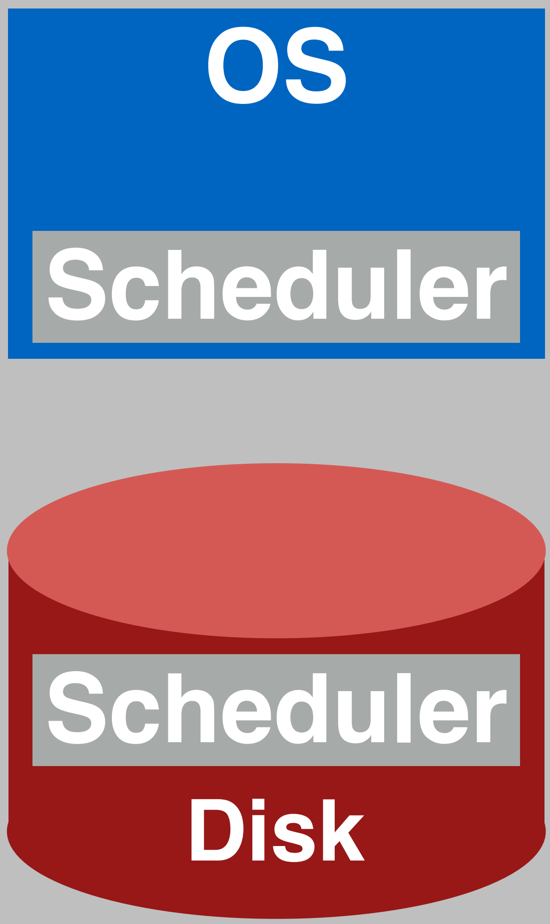
How to implement SPTF?



1. OS has more memory to buffer requests
2. OS has knowledge about active processes

1. Disk knows about exact layout of sectors
2. and precise position of read/write head

How to implement SPTF?



Implication: **Division of work**

1. OS presorts requests, sends limited number to disk

2. "Fine-tuning" on disk

Shortest Positioning Time First

Implementation on **disk**!

Implementation in **OS**?

→ Use **Shortest Seek Time First** (SSTF) instead (ignore rotation, as it is unknown to OS)

Disadvantages of both variants:

- "Greedy": considers only the next decision, not globally optimal
- Far away requests may "starve"

SCAN algorithms

Elevator algorithm

Sweep read/write head back and forth, from one end of disk other, serving requests along the way

- Ignores rotation delays

Pros/Cons?

- No starvation
- Unfair for requests at the extremes: Higher average response time at the edges. **Why?**

C-SCAN

Better: C-SCAN (circular scan)

- Only sweep in one direction
- Then, quickly return to other edge

Advantage:

- Fairer then elevator algorithm: expected response time the same for all sectors

Scenario: What happens?

Two processes each calling **`read()`** with C-SCAN

```
void reader(int fd) {
   char buf[1024];
   int rv;
   while((rv = read(fd, buf)) > 0) {
      process(buf, rv); //takes about 1 ms
   }
}
```

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.**

Completely Fair Queuing (Linux)

- Separate 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)
  → thus **not** work-conserving!

What happens on previous example?

Summary

- For disks, access latencies primarily depend on seek time and rotation time
  - Locality strongly influences throughput
- I/O scheduling different than CPU scheduling:
  - Cost of jobs *known* a priori
  - Cost of jobs strongly *depends on current state*
    - State not completely known at OS level
    - coarse scheduling at OS level
    - finetuning within disk