Today (4/3) ← April (snow)

3 Parts:

~~virtualization~~

~~Concurrency~~

persistence (Input/Output)

Focus: Disk, SSDs + software in OS (file system)

Today:

- → Devices
- → Hard Drives (Hard Disk Drives) HDDs
- → I/O scheduling [Cost]

Break:
- → Project
- → Midterm

Devices :

=) many types

=) hard drive

=) SSD

=) other storage

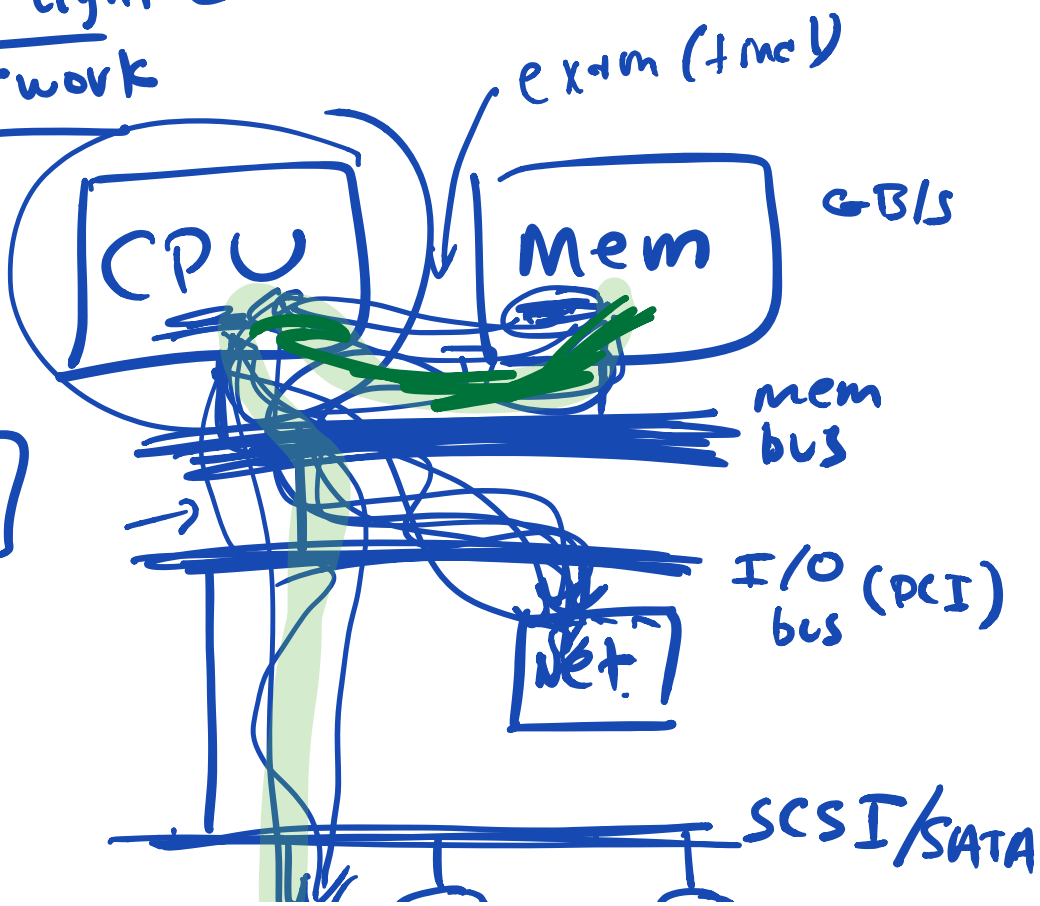
=) keyboard / mouse

=) graphics

=) sound

=) RGB light ⇐ Focus

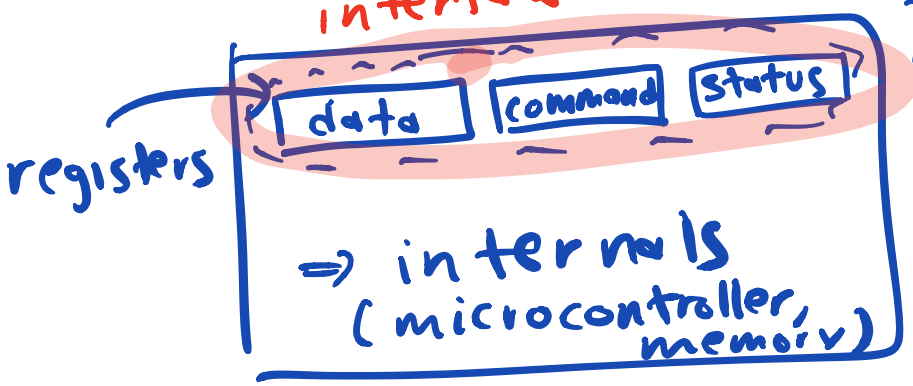
=) Network



Example: (Intel Z270 chipset)

hothardware.com

Device interface: (device controller (hardware that presents interface + implements device functions)



protocol:

```
while (status == BUSY)
    ; // spin (wait)
write data => data register
write cmd => cmd register
while (status == BUSY) // device begins
```

; // spin (wait) ⌟ operation for cmd

Questions:

ignoring: error handling

Inefficiencies

=> spin waiting

=> data movement

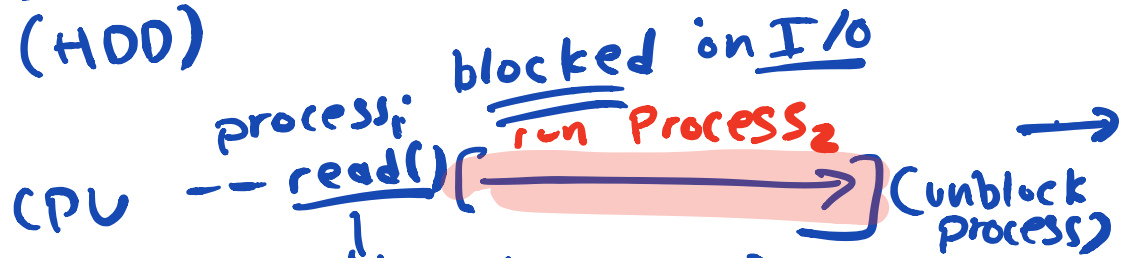
How to do I/O?

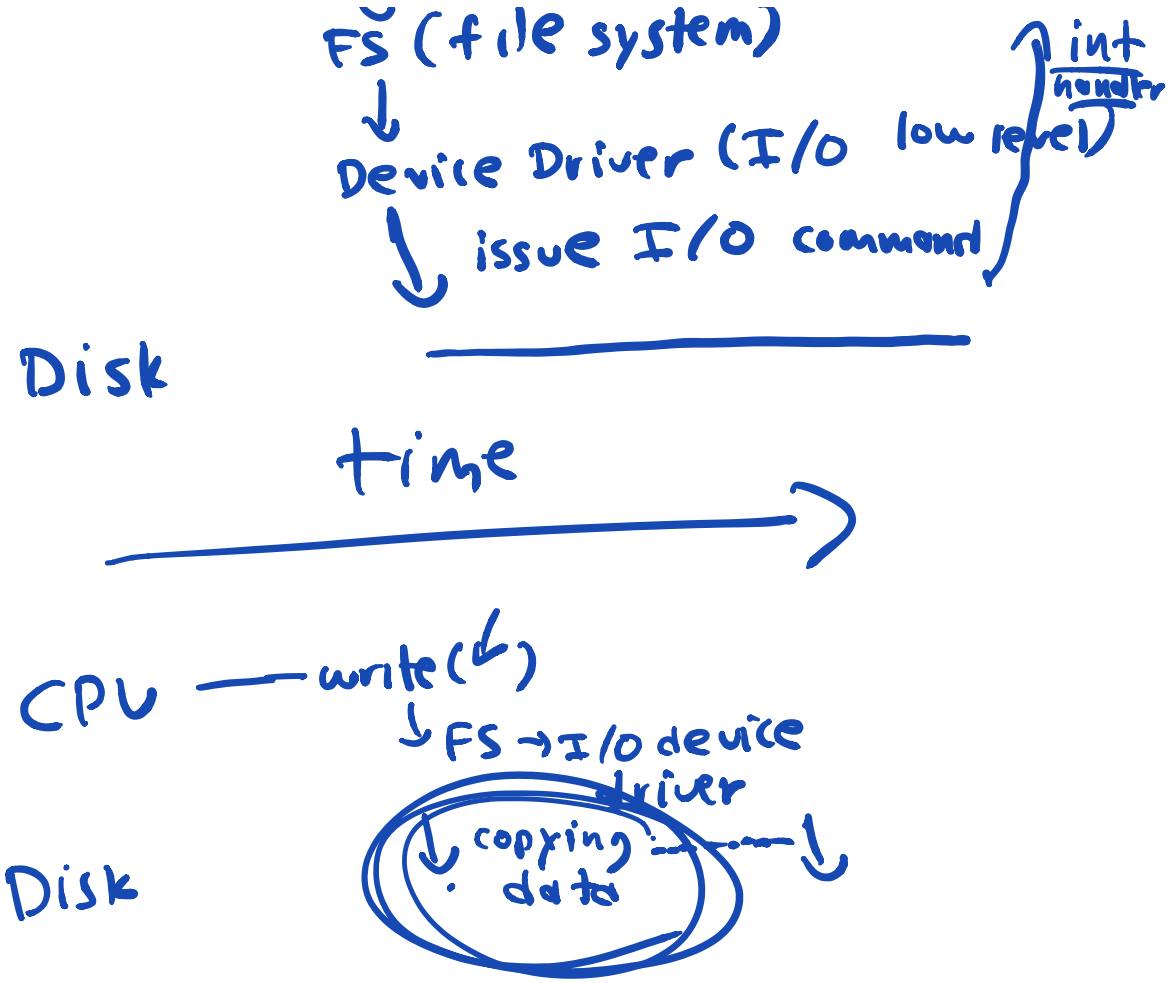
=> how to read/write registers on device?

Spin waiting: Avoid CPU waste (esp. for slow devices)

(polling)

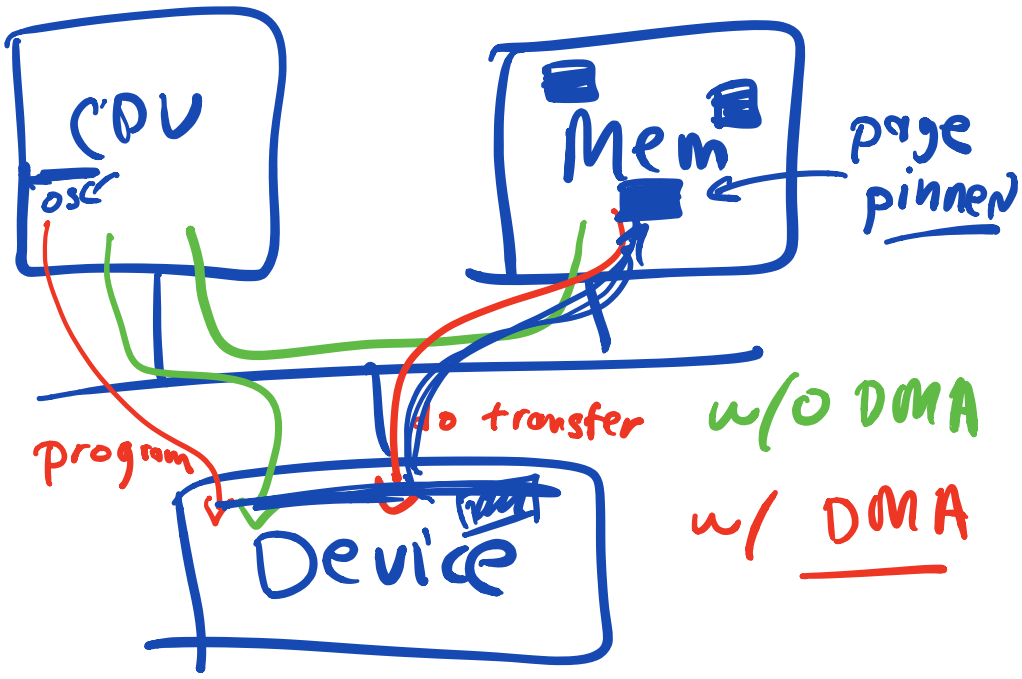
=> Interrupts → overlap





=> reduce CPU burden w/ more specialized hardware:

Direct Memory Access (DMA)



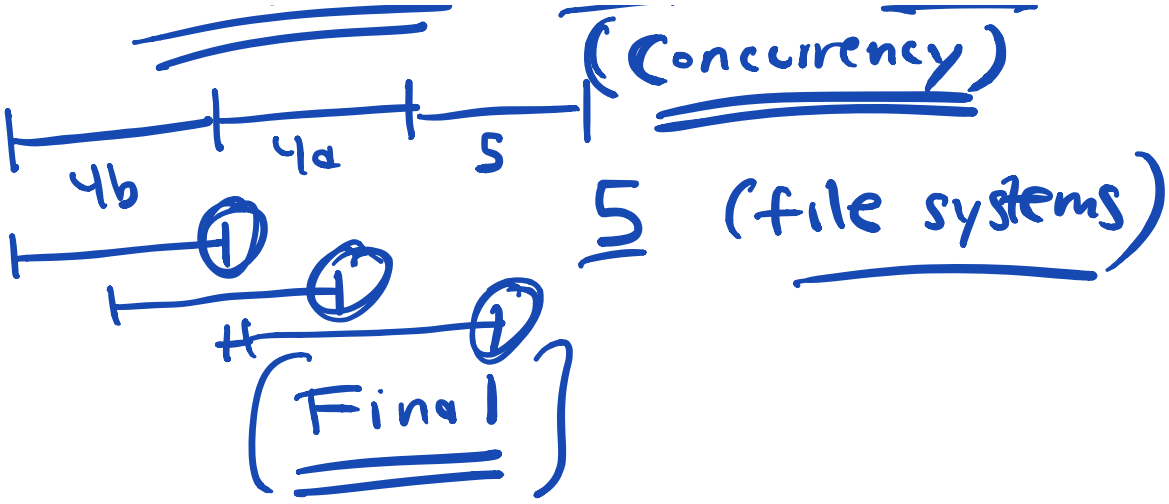
How to interact w/ device?

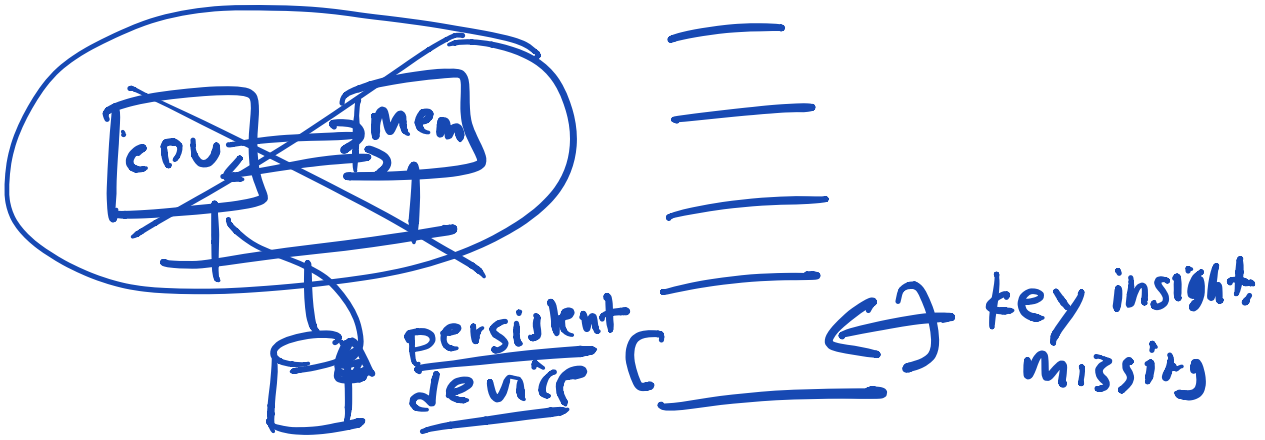
- ⇒ I/O instructions (x86) in/out
- ⇒ memory mapped I/O

Break:

Midterm: info: coming
- answer key
- + your answers

Projects: 4a 4b





Hard Drives: [Interface / Internals]

=> Persistent ~~Storage~~ Device

"CS" perspective:

=> Controller <- interface

↓

control operation

=> Mechanics:
Platters, arm, etc.

Interface: Hard Drive

Read sector, len → return data

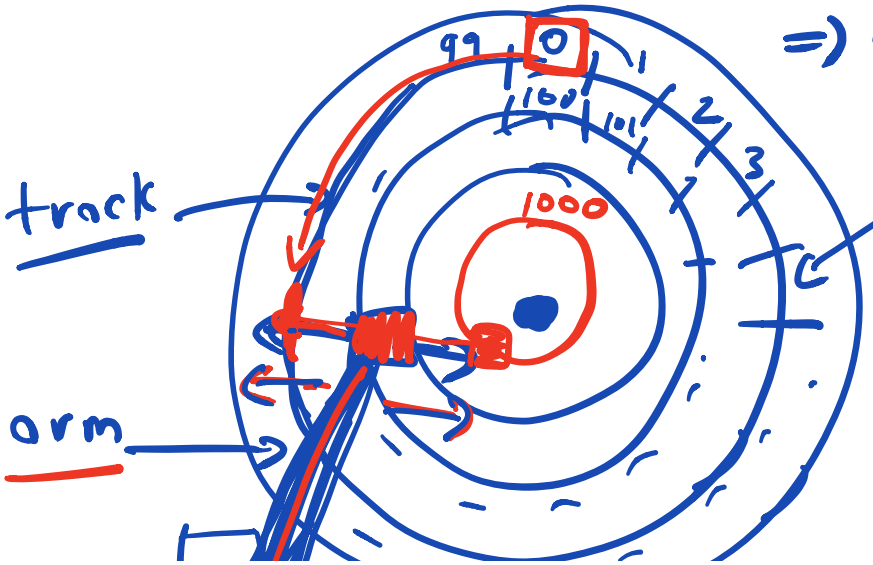
Write sector, len, data → return error code

sectors (512 bytes)

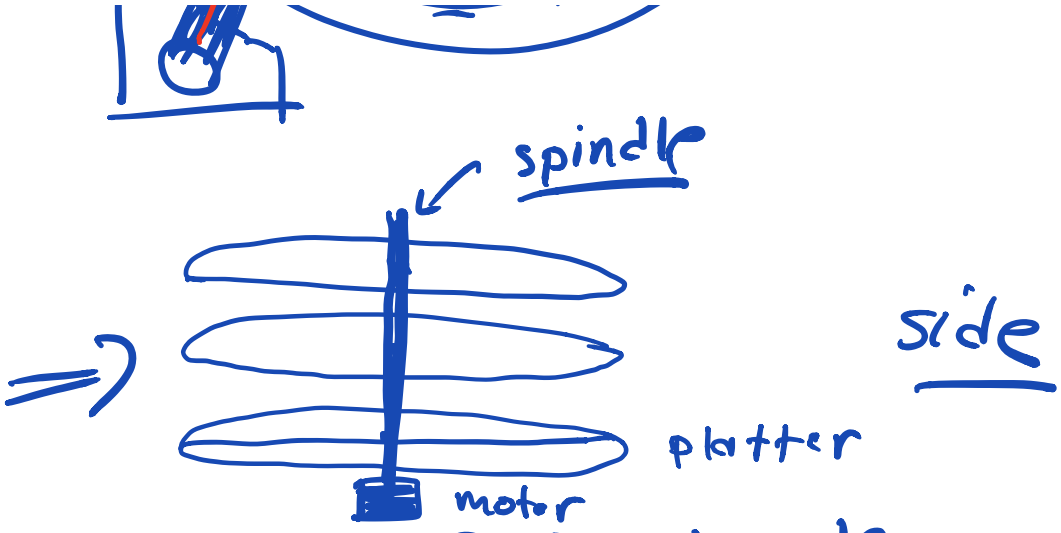
Internals:

mapping: interface => platter, surface, track

=> platter(s)

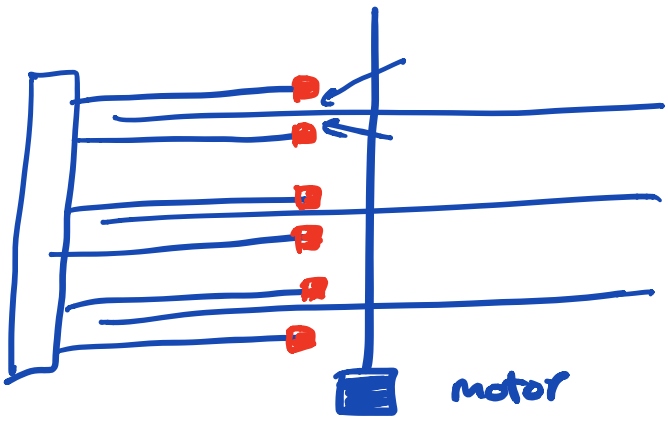


■ : read/write head



runs @ fixed rate

(typical: 7200 RPM, 10K RPM)



Read/write:

Seek: positioning read/write head over correct track

over

rotation (rot. delay): waiting for desired sector to rotate under head

transfer: do read/write

seek rotate transfer

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

large I/Os vs. small I/Os

transfer: 100 MB/s

$seek_{avg}$: 7 ms

rot_{avg}: 3 ms

1 MB I/Os vs. 1KB I/Os

Perf? seek: 7ms, rot: 3ms } 10ms

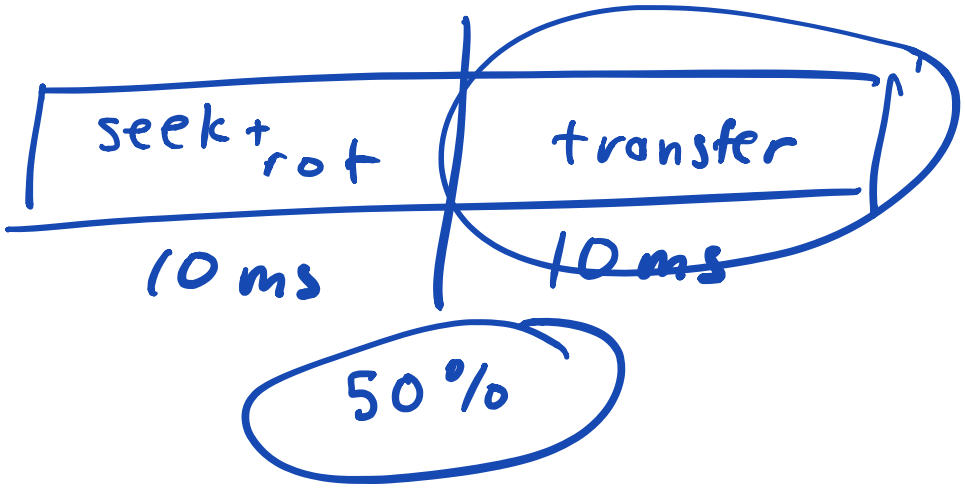
transfer: 10 ms → transfer: ~0

rate

$$1\ \cancel{MB}\left(\frac{1\ \cancel{sec}}{100\ \cancel{MB}}\right)\left(\frac{1000\ ms}{1\ \cancel{sec}}\right) =$$

10 ms

transfer (1 MB)

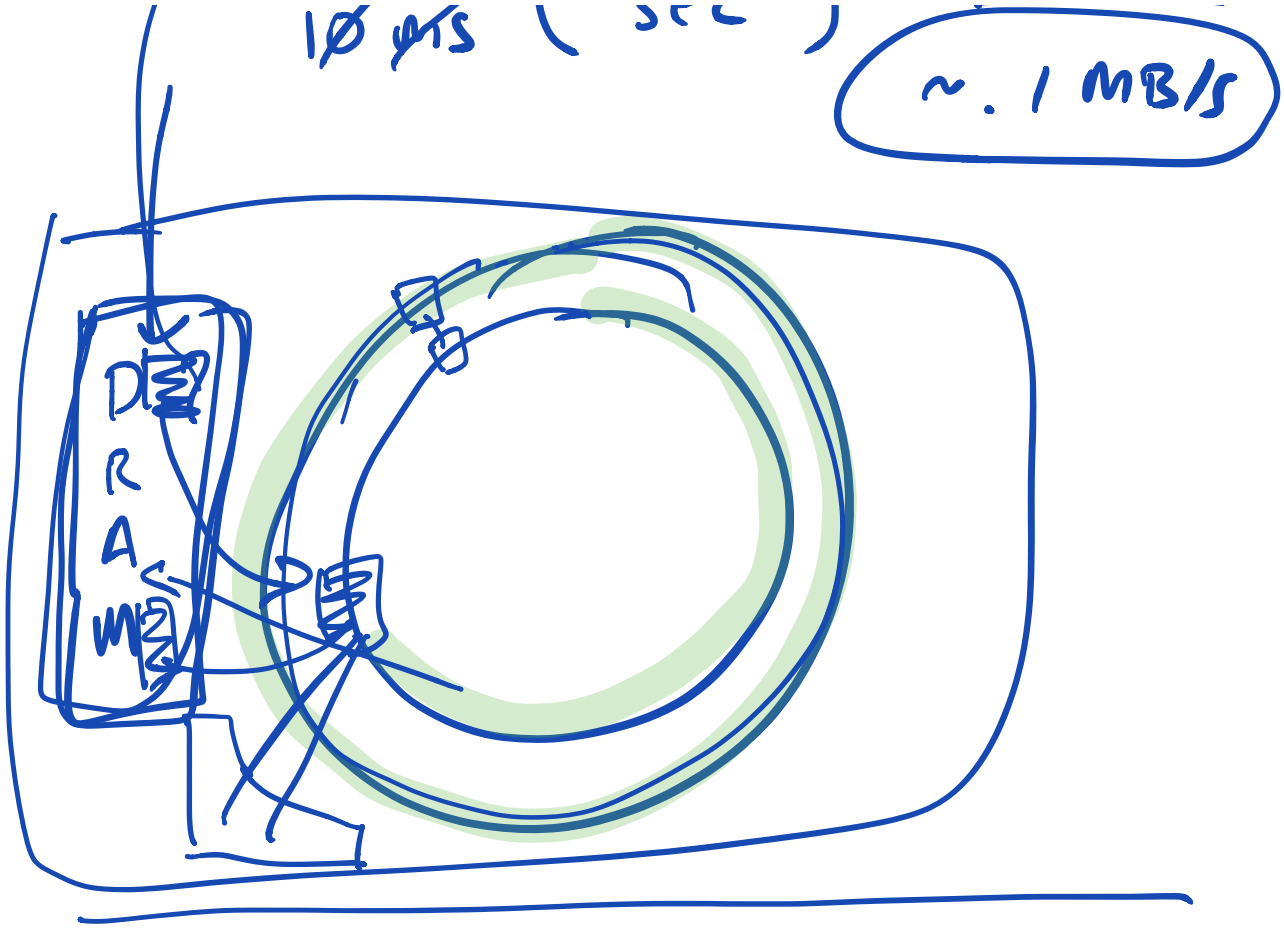


50 %

$$\frac{1\ MB}{20\ ms} \Rightarrow \quad MB/s$$

$$\left(\frac{1\ MB}{20\ \cancel{ms}}\right)\left(\frac{1000\ \cancel{ms}}{1\ sec}\right) = 50\ MB/s$$

$$\uparrow \frac{1\ KB}{\quad}\left(\frac{1000\ \cancel{ms}}{\quad}\right) \Rightarrow 100\ KB/s$$



Break: 3 mins

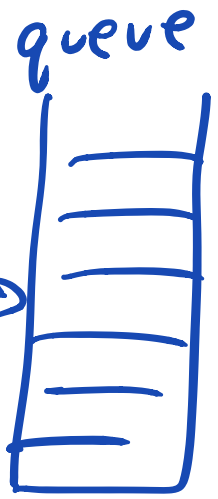
Scheduling (I/O)

=> Given set of requests: have to decide which to do first

Goals:

Fairness

Performance



"Job": I/O request

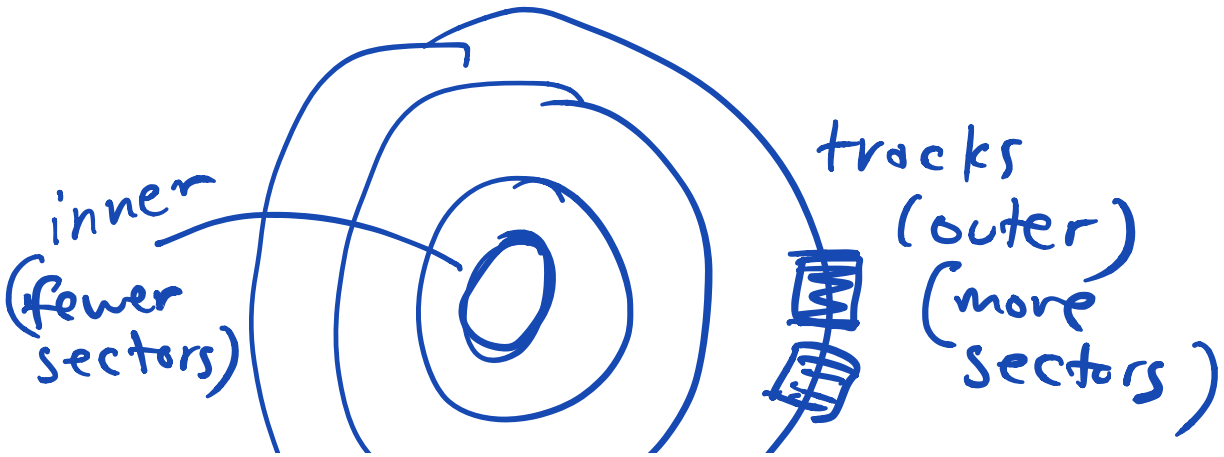
desire: shortest job first

need to know: how long it runs for

here: (kind of) possible

OS: but hard drive internals hidden

inside drive: have much more knowledge



Sched:

FIFO: X

Shortest Seek Time First (SSTF)

→ Starvation

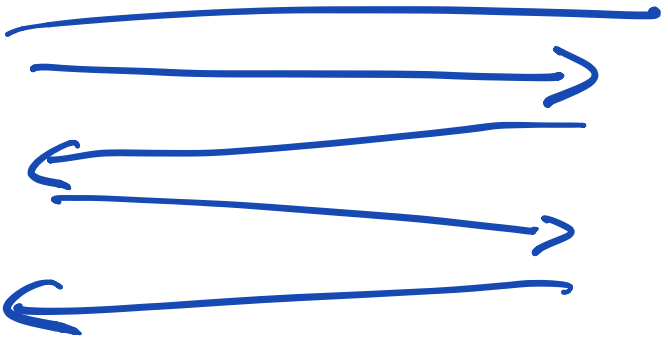
→ no accting of rotate

→ Shortest Access Time First (SATF)

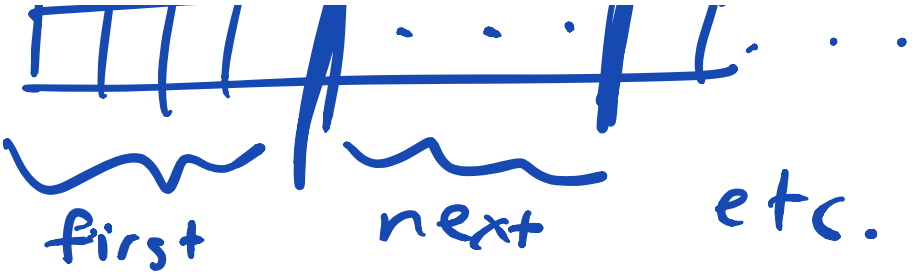
(rot + seek)

→ Solving starvation:

"SCAN" (elevator)



→ Bounded Queue



Is SATF best?
(optimal)