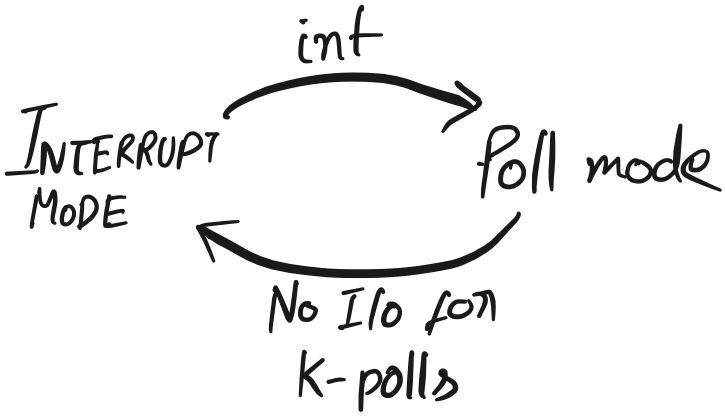
CS202: I/O, Disks

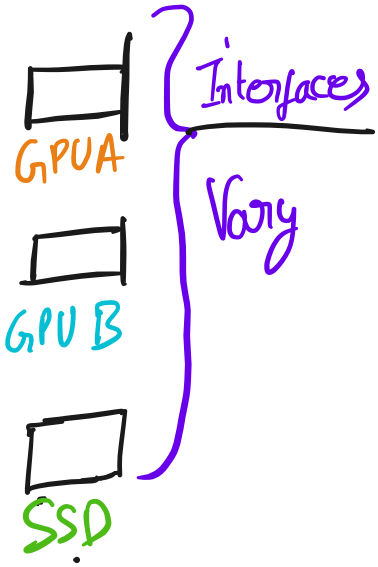
Where we were

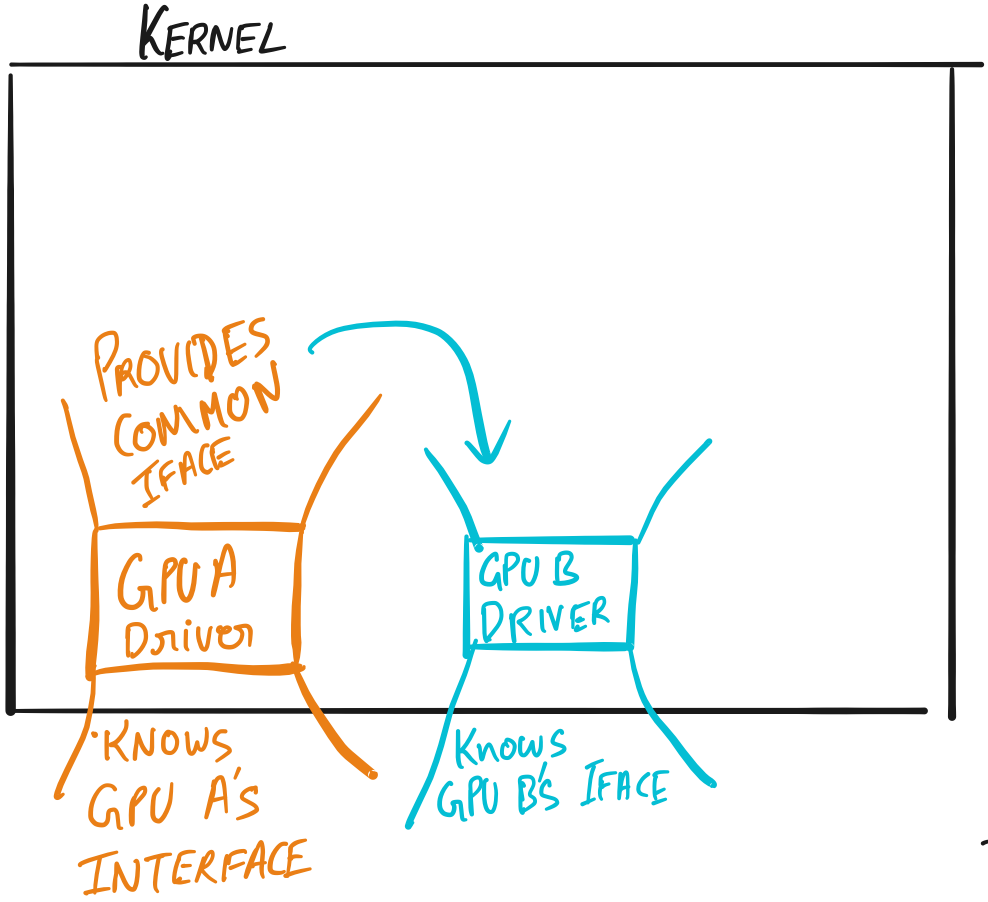
- Transfer data to/from devices
  - I/O instructions inb/inw/outb/outw
  - Memory mapped I/O
  - DMA
- Coordination w/devices
  - Interrupts
    - Don't waste cycles polling
    - High interrupt rates lead to no FORWARD PROGRESS
  - Polling
    - Might waste cycles
    - OS decides when to check
      ↳ FORWARD PROGRESS
  - IN PRACTICE

ADAPTIVE



DEVICE DRIVERS





SYNCHRONOUS VS ASYNCHRONOUS I/O

↳ FROM USERSPACE

So far: read/write/... block the calling thread

Assumption: Instead has nothing
do while waiting

But, what if thread could do other things
(e.g., switch to & run different
usermode thread)

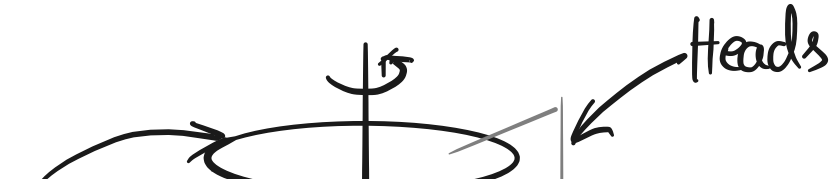
Most operating systems offer non-blocking APIs
Not standard

Do not need to know
- Linux: epoll, io-uring
- OSX: kqueue
- Windows: Completion ports

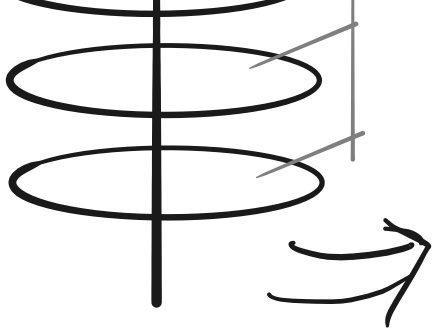
Similarity
→ Poll for ~~completion~~
→ Mechanism to wait/block if necessary

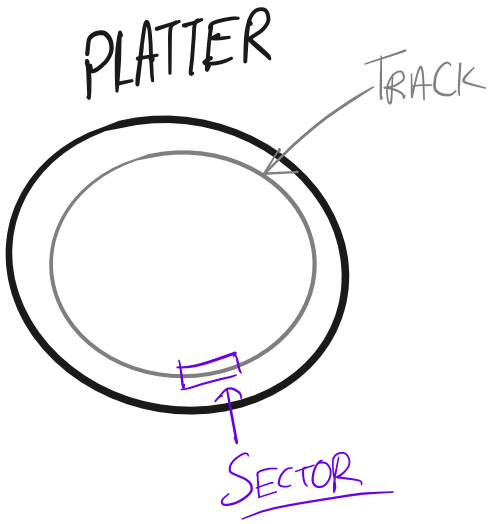
Disks: Spinning Disks

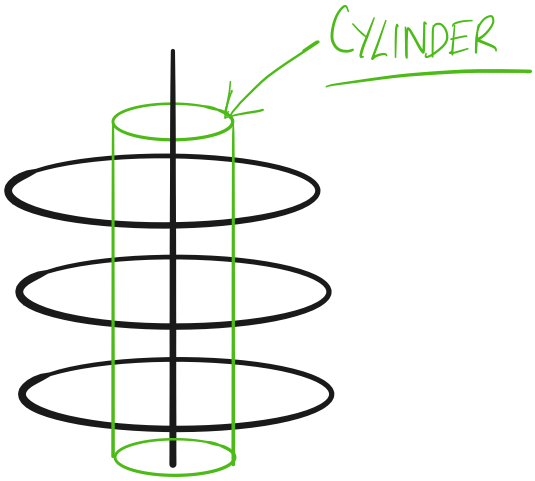
Why? - Still widely used
- Dictate the design of most/many file systems



Platter (1-8)







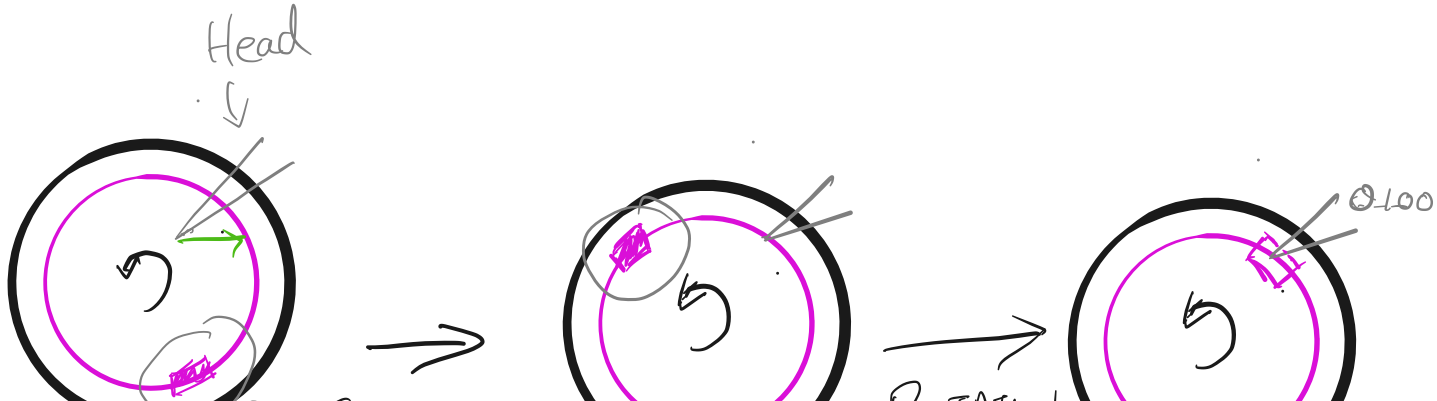
Interface: Linear array of sectors (generally 512 bits)

READ/WRITE 4K starting from

Read(25)

How: DISK CONTROLLER (Implemented by disk hardware) translates sector → <platter, track, sector>

Read(25)



Tot. TRANSFER TIME = ROTATIONAL DELAY + SEEK DELAY + TRANSFER TIME

ROTATIONAL DELAY
↳ DISK ROTATION
E.g. 7200 RPM
10 000 RPM

SEEK DELAY
↳ Average seek delay
E.g. 8-10ms

TRANSFER TIME
↓
Throughput
E.g., 100 MB/s
SIZE/TPUT = Time

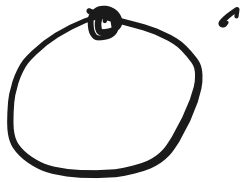
⇓

Avg. Rotation time from RPM

7200 RPM ⇒ 120 RPS

⇒ 1 Rotation every

$1/120$ s ≃ 8.3ms per rotation



⇒ Avg Rotation time = 4.15ms

10000 RPM ⇒ ~ 167 RPS

⇒ ~ 6 ms per rotation

⇒ Avg Rotation time = 3ms

Let us use this:

Disk: Rotation 12,000 RPM

Avg. Seek Time 12ms

Transfer rate 128 MB/s

Assume 512 Byte reads ⇒ 512 B sectors

ⓐ Throughput (bytes/second) to read 500 sectors spread randomly across disk, served in FIFO order?

500 · 512B / 500 · T

Read 27, 12, 100, 17, ...

Might need to seek for each sector

Tput = Data Read / Time taken.

Data read = 500 · 512 B

Time taken for 1 read (on avg)

Seek time = 12 ms

Rotation time = 2.5ms

12000 RPM = 200 RPS

$\Rightarrow$ 1 rotation every $1/200$ s = 5ms

$\Rightarrow$ Avg Rot time = 2.5ms

1MB = 2^{10} KB = $2^{10} \cdot 2^{10}$ B

$$\text{TRANSFER Time} = \frac{512\,B}{128\,MB/s} = \frac{2^9}{2^7 \cdot 2^{10} \cdot 2^{10}}\,s$$

$$= \frac{1}{2^{18}}\,s \simeq 4\mu s$$

Time for 1 TRANSFER $\simeq$ 12ms + 2.5ms + .004ms $\simeq$ 14.5ms

$$\text{TPUT} = \frac{500 \times 512B}{500 \times 14.5ms}$$

$$= \frac{512\,B}{14.5\,ms} = \frac{512\,000\,B}{14.5\,s}$$

$$\approx 34.5\,KB/s$$

(b) Throughput for 500 sequential sectors?

5, 6, 7, ... 505

What changes:- Seek once to track, wait once for head to arrive at 5.

Seek time = 12ms

Rotation time = 2.5ms

Transfer time = $\frac{500 \cdot 2^9\,B}{2^7 \cdot 2^{20}\,B/s} = \frac{500}{2^{18}}\,s$

$$\approx \frac{2^9}{2^{18}}\,s = \frac{1}{2^9}\,s = \frac{1}{512}\,s$$

$$\approx 2\,ms$$

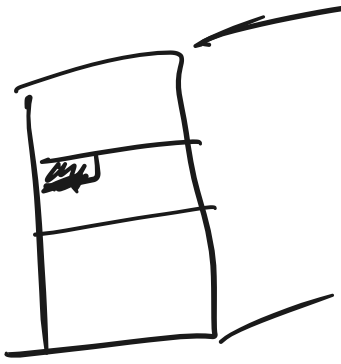
$$Tput = \frac{500 \cdot 2^9\,B}{16.5\,ms} = \frac{500 \cdot 10^3 \cdot 2^9}{16.5}\,B/s$$

$\sim$ 18 MB/s

Observation: Sequential Access Is Much Faster

↳ Constraint For A Lot Of File System Design.

· write (



Read

write.