PERSISTENCE: DISK SCHEDULING

Shivaram Venkataraman

CS 537, Spring 2019

Administrivia

Grades: Project 2b, 3, midterm grades out!
See Piazza for regrade information

Project 4a is out! Due April 4^{th}
More details in discussion section

Out of town Monday, Tue next week.
Guest lecture on Tuesday

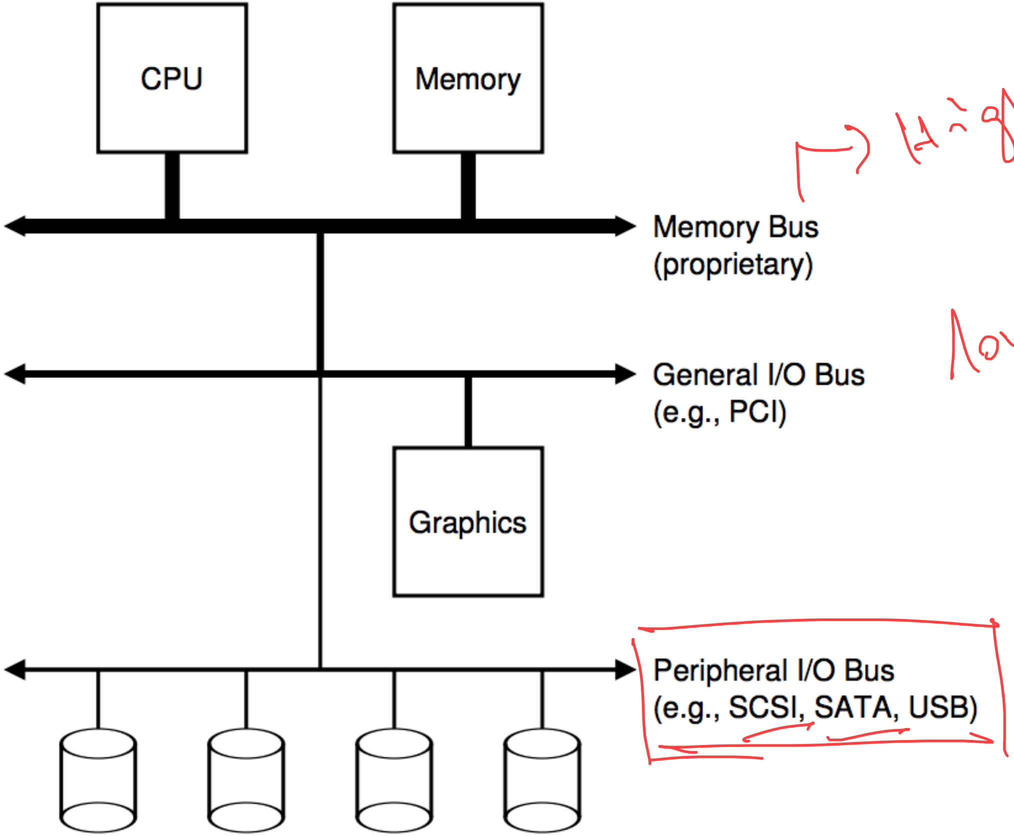
AGENDA / LEARNING OUTCOMES

How do you calculate sequential and random tput of a disk?

What algorithms are used to schedule I/O requests?

RECAP

HARDWARE SUPPORT FOR I/O



High speed

lower speed

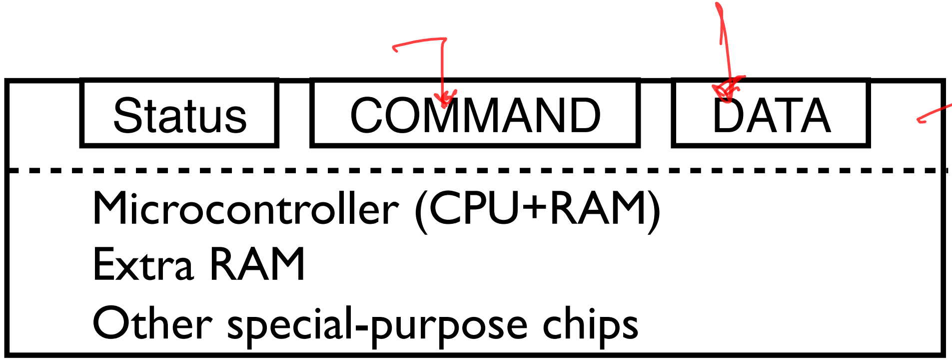
greater distance

hard drive

3.5 inch

SSD, keyboard

EXAMPLE WRITE PROTOCOL



Registers
"read / write"

waiting

Check if device is not busy

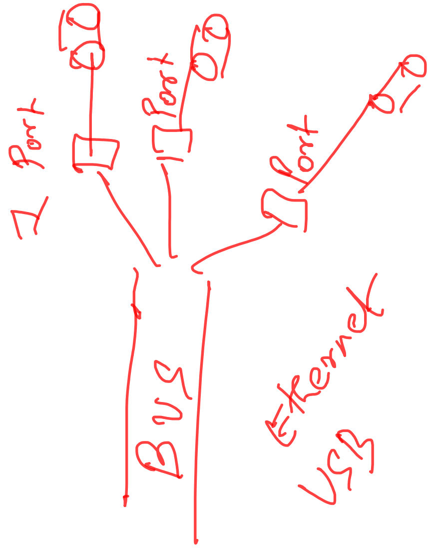
```
while (STATUS == BUSY)
   ; // spin
Write data to DATA register
Write command to COMMAND register
while (STATUS == BUSY)
   ; // spin
```

Copy

waiting

wait for the operation

PROTOCOL VARIANTS



/dev/

Status	COMMAND	DATA

Microcontroller (CPU+RAM)
Extra RAM
Other special-purpose chips

Status checks: polling vs. interrupts

CPU do other work

Trade-offs
very fast device ⇒ polling

PIO vs DMA

CPU stall avoided "DMA"

I/O

Special instructions vs. Memory mapped I/O

IN/OUT

LOAD/STORES
special registers

Programmable

HARD DISK INTERFACE

Disk has a sector-addressable address space

Appears as an array of sectors

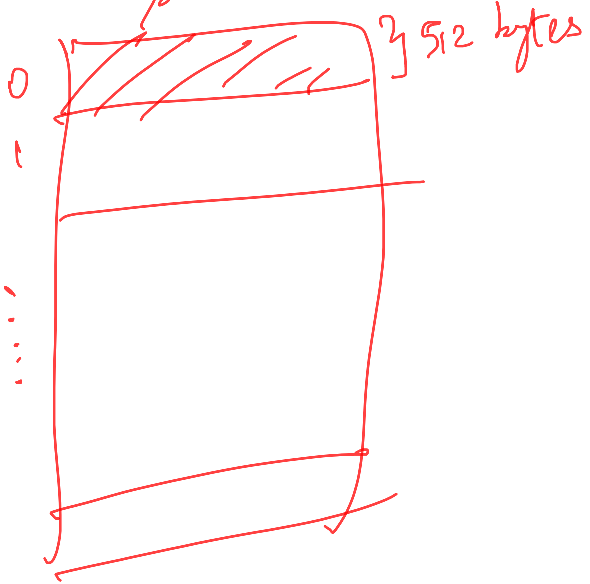
"Page" in memory

Sectors are typically 512 bytes

Main operations: reads + writes to sectors

Atomic

Mechanical and slow (?)

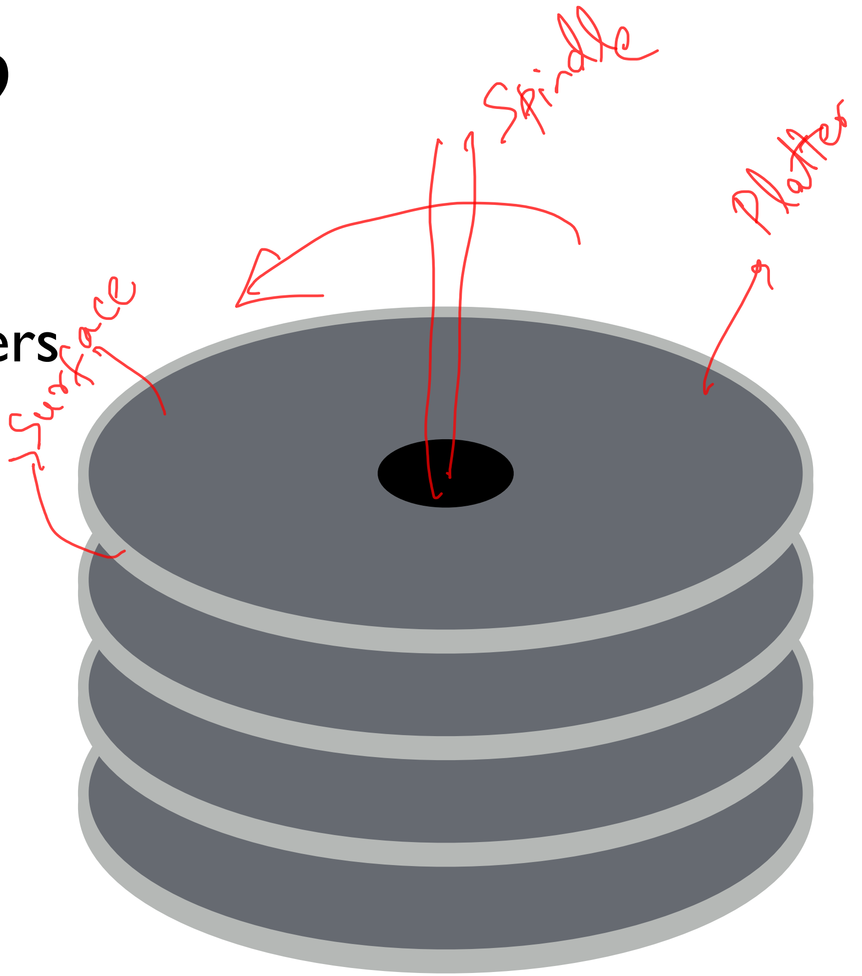


RPM?

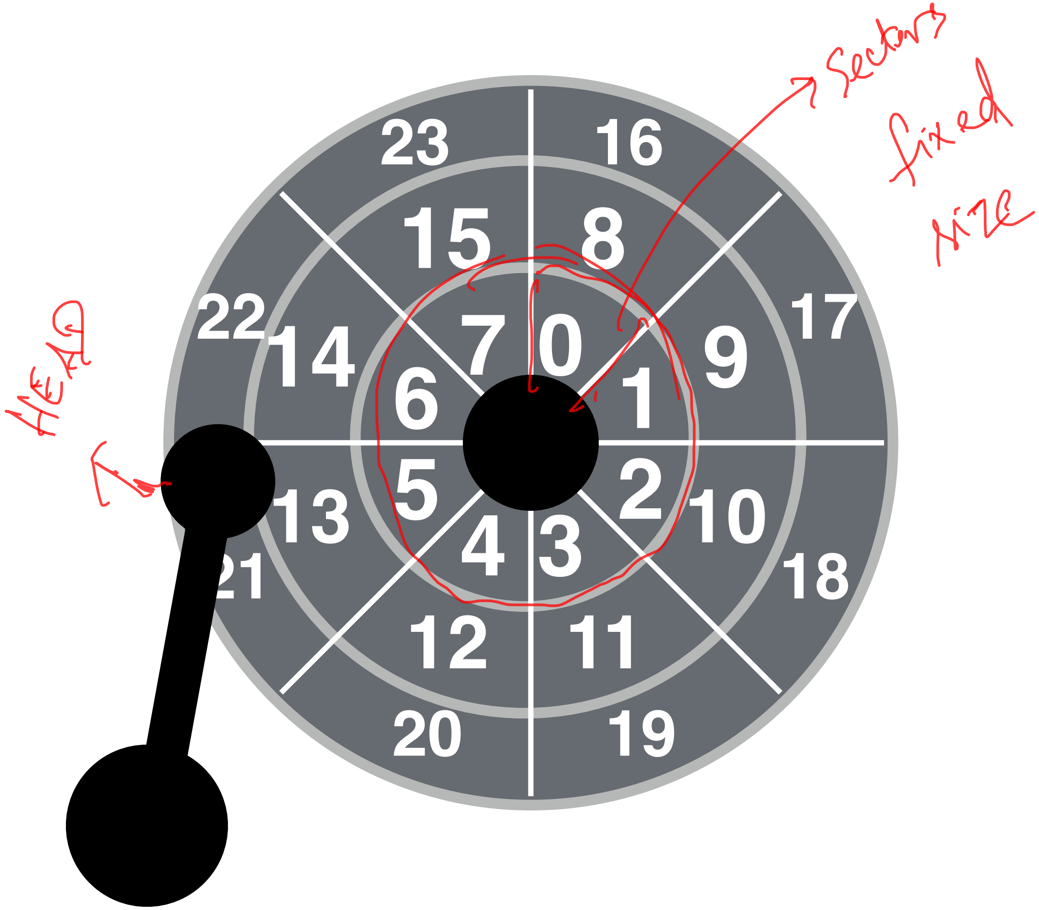
Motor connected to spindle spins platters

Rate of rotation: RPM

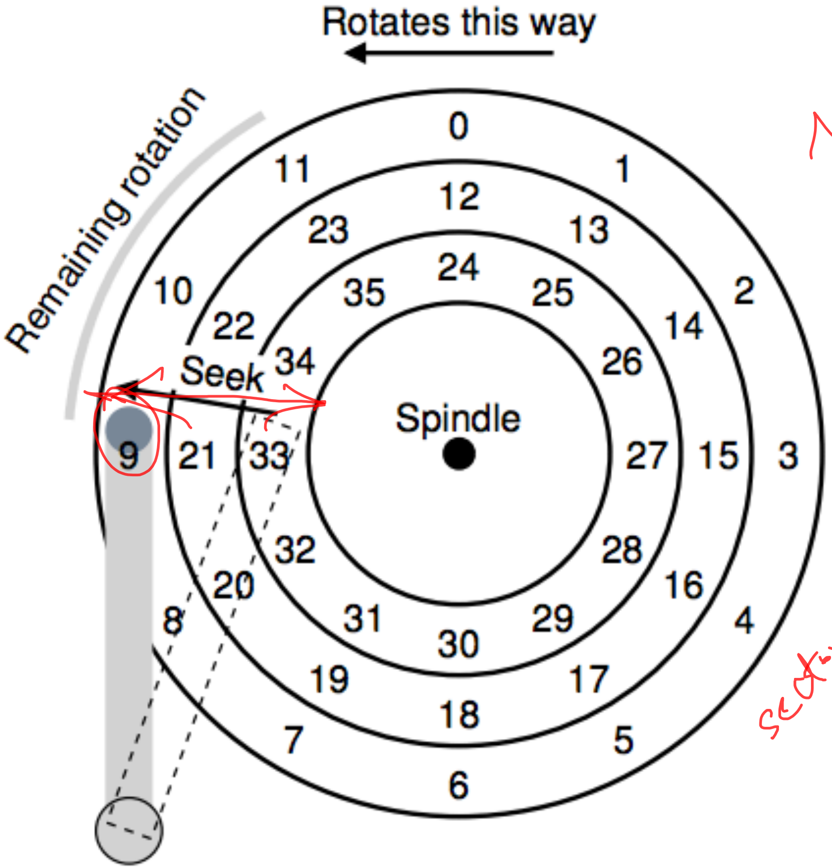
10000 RPM → single rotation is 6 ms



Heads on a moving arm can read from each surface.



READING DATA FROM DISK



1 Tb disk

Num surface

sector size = 512 bytes

Seek Time = Head right track

Rotational delay = Surface rotate right sector is below the Head

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 $\boxed{4 - 10 \text{ ms}}$

Average seek = 1/3 of max seek

Depends on rotations per minute (RPM)

- 7200 RPM is common, 15000 RPM is high end

Average rotation?

Half the max rotation time

Pretty fast: depends on RPM and sector density.

100+ MB/s is typical for maximum transfer rate

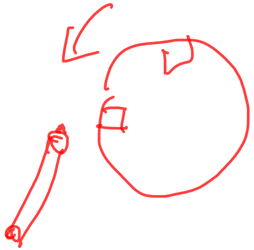
BUNNY 11

What is the time for 4KB random read?

High Perf

Capacity

	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



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

	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

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

$T_{seek} = 4\ ms$

$T_{rotate} = \frac{1\ \cancel{15000}\ R}{4\ 60 \times \cancel{1000}\ ms} = \frac{1}{4}$ Rot per ms $\Rightarrow$ 4 ms / rot

$\Rightarrow$ 2 ms on avg

$T_{transfer} = \frac{4\ KB}{125\ MB} \cdot s = \frac{4\ KB}{125 \times 10^3\ KB} \cdot 10^3\ ms = \frac{4}{125}\ ms = 0.032\ ms$

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

	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

$T_{seek} = \boxed{9 \text{ ms}}$

$T_{rot} = \frac{72\cancel{00} \text{ RPM ms}}{60 \times 10\cancel{00}} = \frac{72 \text{ rpms}}{600} = \frac{600}{72} \text{ ms/rot} = 8.33$

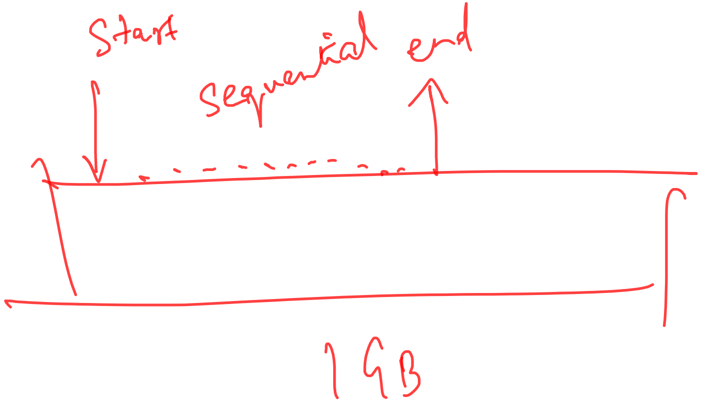
$\text{Avg} = \boxed{4.16 \text{ ms}}$

$T_{transfer} = \frac{4}{105} = \boxed{0.038 \text{ ms}}$

$= 13.2 \text{ ms}$

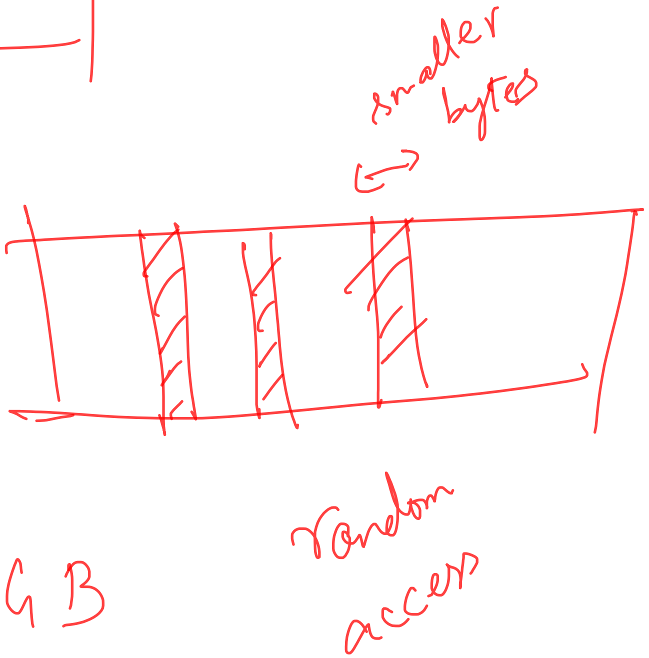
WORKLOAD PERFORMANCE

WORKLOAD PERFORMANCE



So…

- seeks are slow
- rotations are slow
- transfers are fast



How does the kind of workload affect performance?

Sequential: access sectors in order

Random: access sectors arbitrarily

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?

100 MB

Cheetah = 4ms + 2ms + ~~1s~~ ≃ 1s ≈ 1.006 · effective tput 125 ≃ MB/s

random 4KB = 6ms ⇒ 4KB/6ms <<< 125MB/s

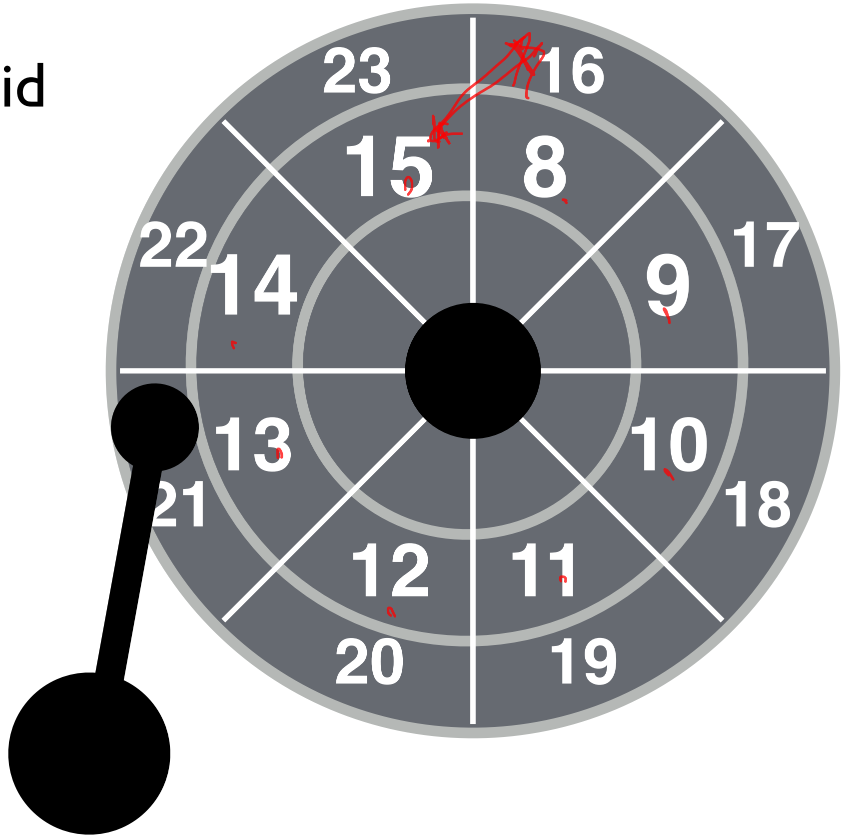
OTHER IMPROVEMENTS

Track Skew

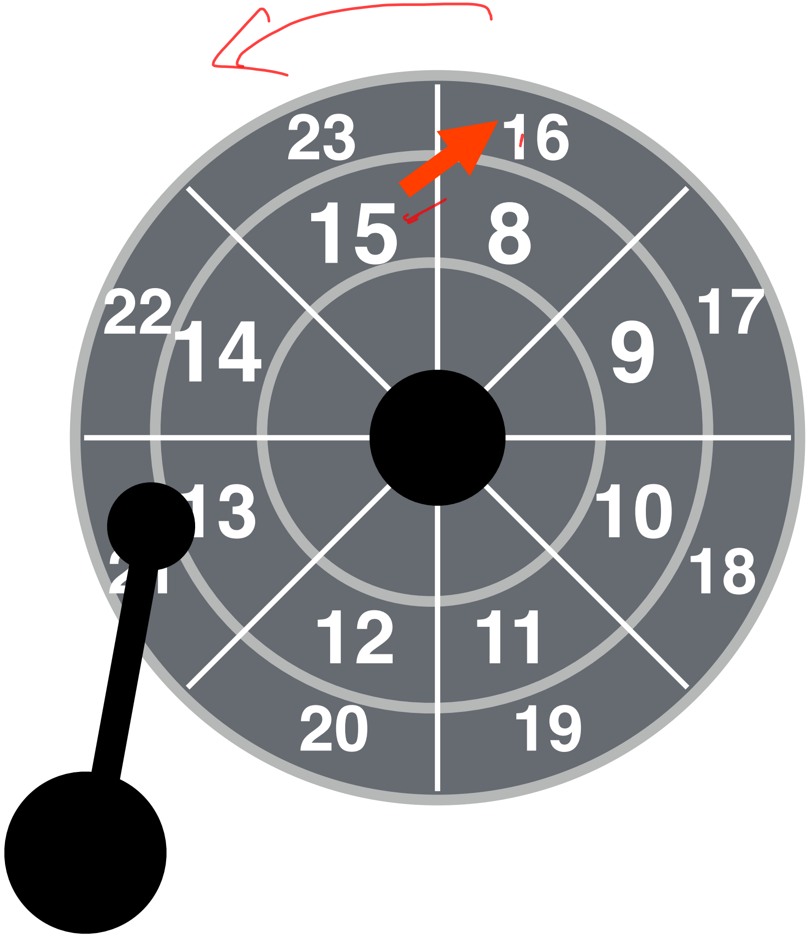
Zones

Cache

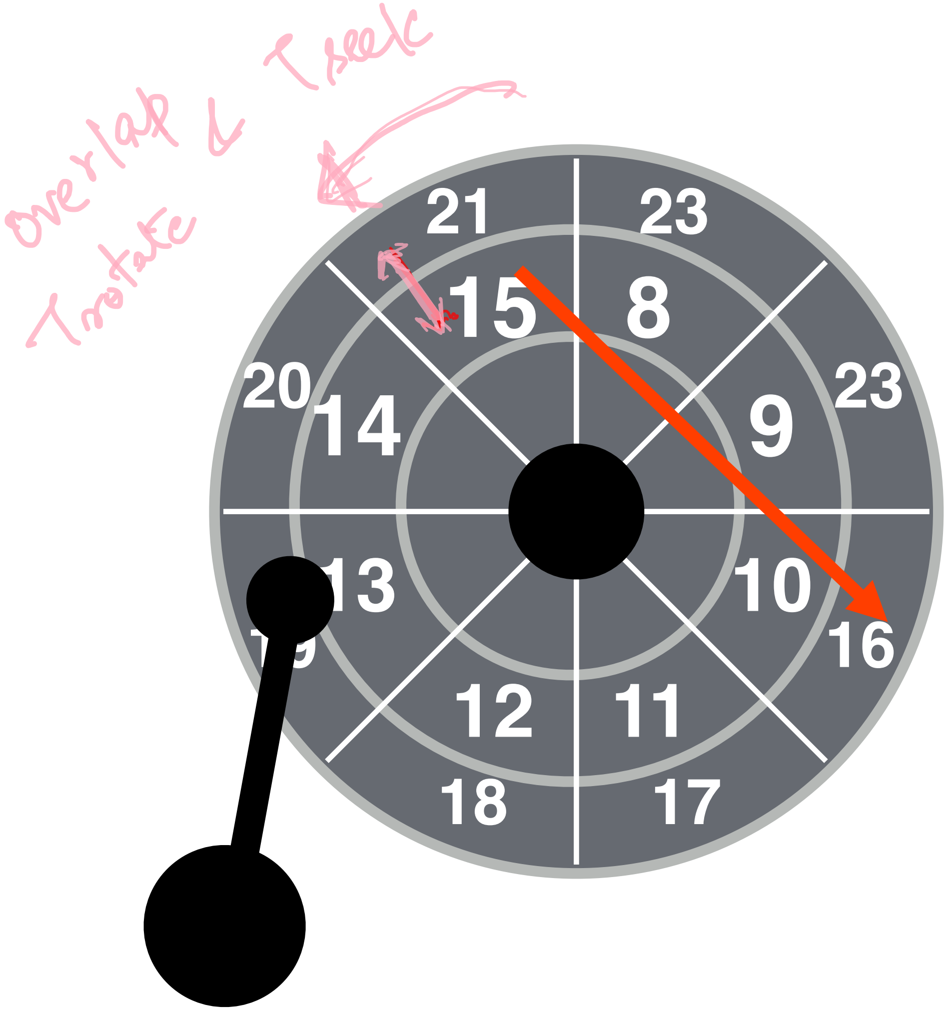
Imagine sequential reading,
how should sectors numbers be laid out on disk?

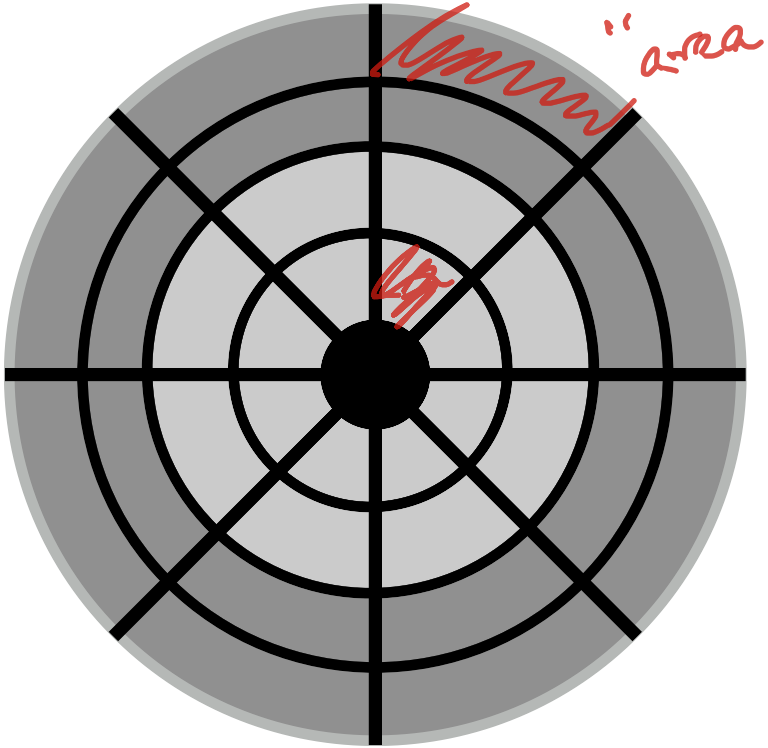


When reading 16 after 15, the head won't settle
quick enough, so we need to do a rotation.



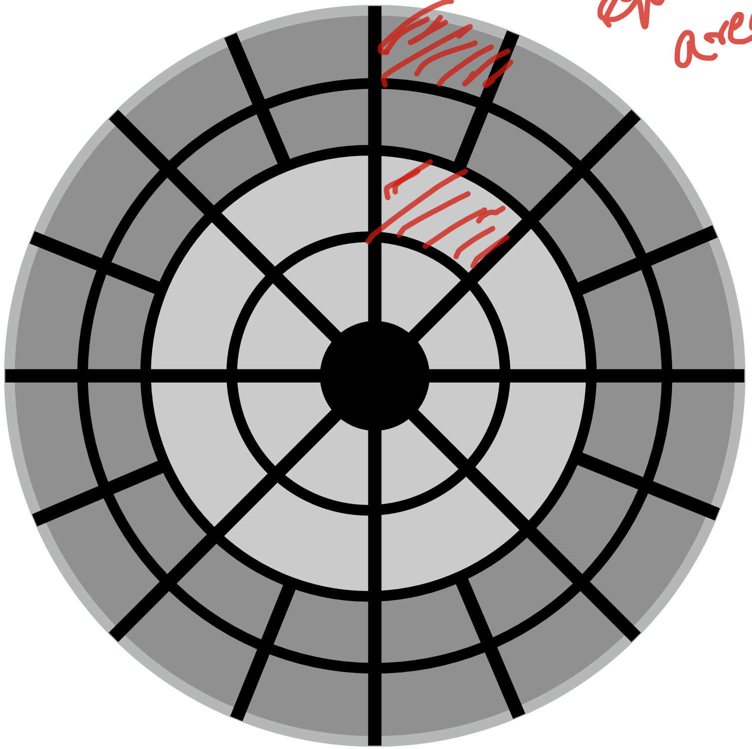
Track Skew





Storage
"area"

outer tracks
more area



equal area

⇒ more sectors in outer tracks

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

OTHER IMPROVEMENTS

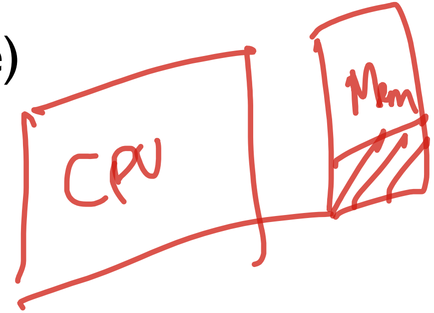
Track Skew

Zones

Cache

DRIVE CACHE

Drives may cache both reads and writes. (In addition to OS cache)



What advantage does caching in **drive** have for reads?

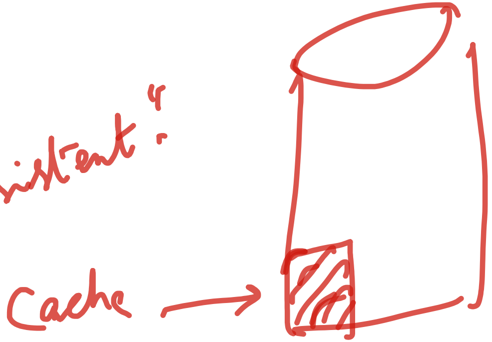
Store recently read sectors. Fetch it from cache

Read ahead

What advantage does caching in **drive** have for writes?

CPU doesn't need to wait for write to finish.

Acknowledge before "persistent".



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

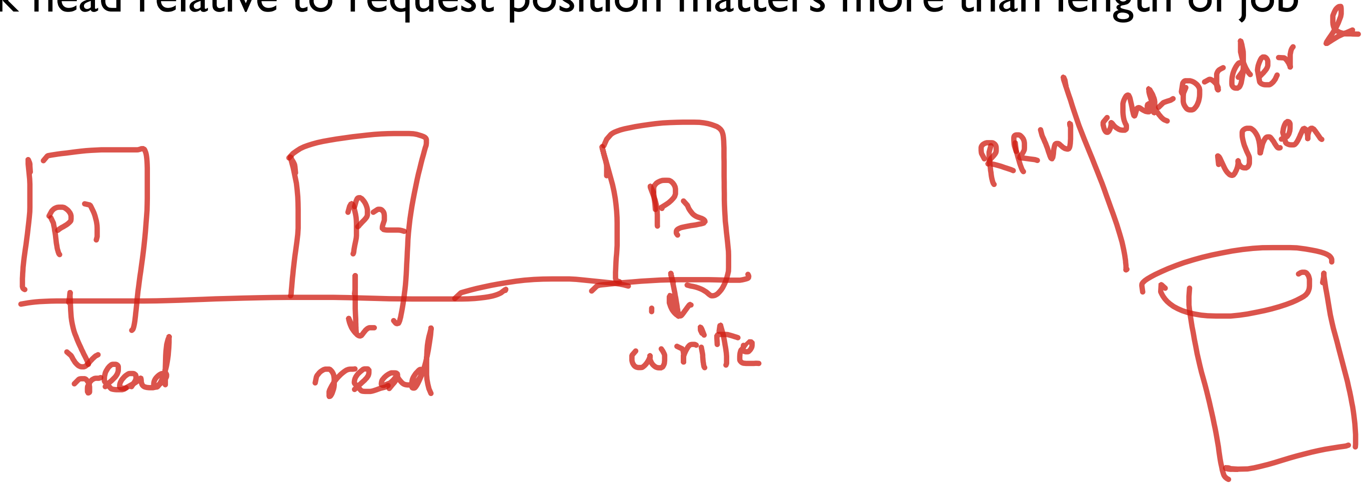
I/O SCHEDULERS

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



BUNNY 12

https://tinyurl.com/cs537-sp19-bunny12

FCFS (FIRST-COME-FIRST-SERVE)

https://tinyurl.com/cs537-sp19-bunny12

FIFO

Assume seek+rotate = 10 ms for random request

How long (roughly) does the below workload take?
Requests are given in sector numbers

10ms 10ms 10ms

300001, 700001, 300002, 700002, 300003, 700003

60 ms

long 10ms

300001, 300002, 300003, 700001, 700002, 700003

20 ms

Why I/O scheduling

3x

SSTF (SHORTEST SEEK TIME FIRST)

next best req

minimize seek

30001 →

Strategy always choose request that requires least seek time
(time for seeking and rotating)

Greedy algorithm (just looks for best NEXT decision)

How to implement in OS?

↳ sector number "substitute" nearest sector number

Disadvantages?

Starvation

30002, 30003 . . . . . .

$\underbrace{\text{70001}}_{\text{starved}}$

101, 102, 103, 104 ... 108, 201, ... 109

SCAN

Input 101, 201, 102, 301, 203

order 101, 201, 301, 203, 102

SCAN or 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

"Best effort" → "sorting"

C-SCAN (circular scan): Only sweep in one direction

Pros/Cons?

"Fair"

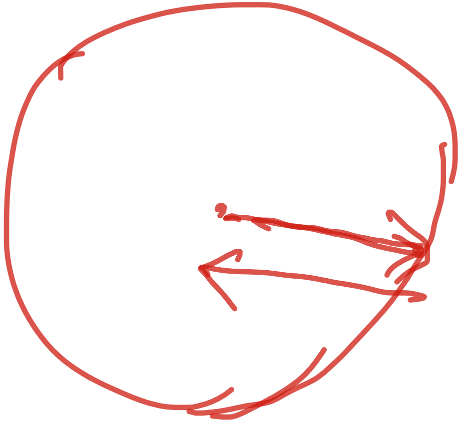
1 → 2 → 3

1 → 2 → 3

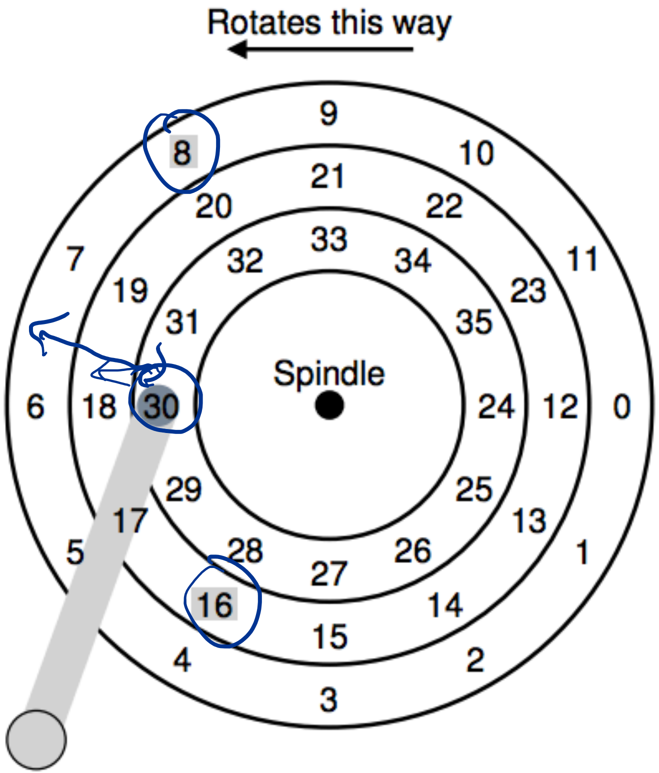
1 → 2 → 3

3 → 2 → 1

1 → 2 → 3



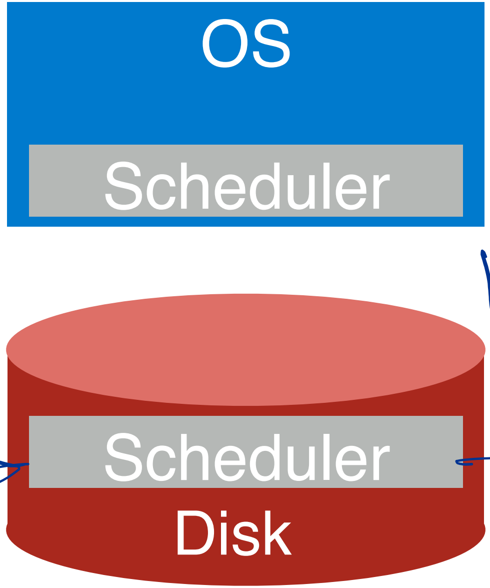
SPTF (SHORTEST POSITIONING TIME FIRST)



SATF (SHORTEST ACCESS TIME FIRST)

SCHEDULERS

Knows process
fair sharing across
software ⇒ upgrade



Sorts request
sector location

|| multiple

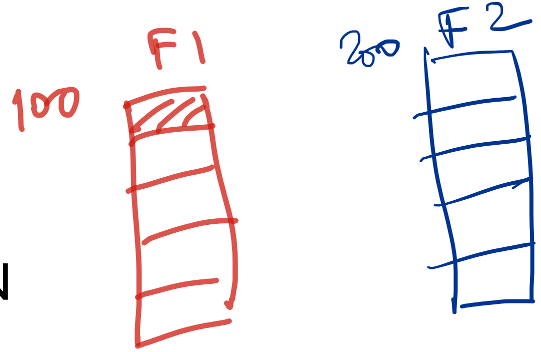
Where should the scheduler go?

Hardware

Knows
disk
geometry

seek time
rotation time

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);
    }
}
```

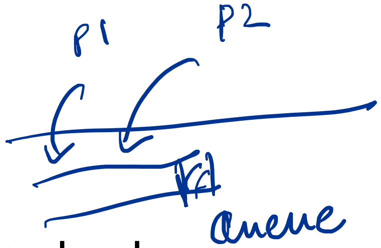
fd,

100, 101 200 201 102, 103, 202...

1ms

request

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

Possible improvements from I/O Merging

Work conserving

Always run a request if resource is free

Not work conserving

Wait for a while to "merge" or get a better sequence

SUMMARY

Disks: Specific geometry with platters, spindle, tracks, sector

I/O Time: rotation_time + seek_time + transfer_time

Sequential throughput vs. random throughput

Advanced Techniques: Skewed layout, caching

Scheduling approaches: SSTF, SCAN, C-SCAN

Benefits of violating work conservation

DISK SIMULATOR HOMEWORK

Rotational speed is set to 1 degree per time. Complete revolution takes 360 time

Transfer begins and ends at the halfway point between sectors. E.g., to read sector 10, the transfer begins halfway between 9 and 10, ends halfway between 10 and 11.

There are 12 sectors per track, meaning that each sector takes up 30 degrees.
To read a sector, it takes 30 time units (given our default speed of rotation).

Disk head is positioned on the outside track, halfway through sector 6.

Compute the seek, rotation, and transfer times for the following sets of requests:

1. -a 7
2. -a 7,30,8
3. -a 10,11,12,13

```
python disk.py –a <cmd> -G
```

Compare FIFO and SSTF for request stream 7,30,8

```
python disk.py –a 7,30,8 -p <SSTF|FIFO>
```

NEXT STEPS

Next class: How to achieve resilience against disk errors

Project 4a in Discussion today

Guest lecture on Tuesday