Hello!

PERSISTENCE: DISK SCHEDULING

Shivaram Venkataraman

CS 537, Spring 2023

ADMINISTRIVIA

Project 4 grades out. Regrades? → Keren Chen

Project 5 – due soon?

→ Tuesday

Midterm 2 – April 4th, lots of details on Piazza

↳ Venue, Time, Videos Old Exams

Thursday - guest lecture

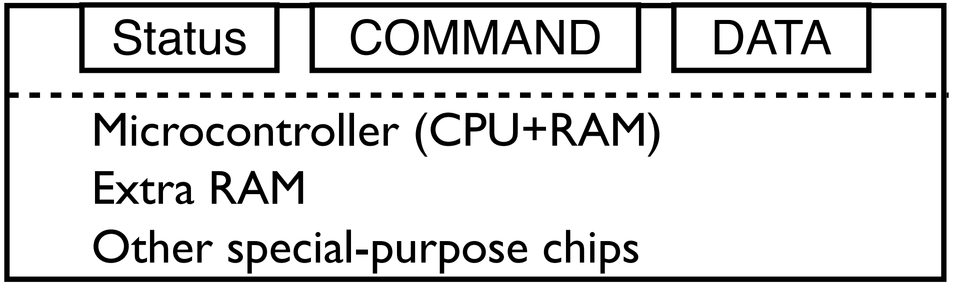
AGENDA / LEARNING OUTCOMES

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

What algorithms are used to schedule I/O requests?

RECAP

EXAMPLE WRITE PROTOCOL




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

DMA

Polling vs. Interrupt

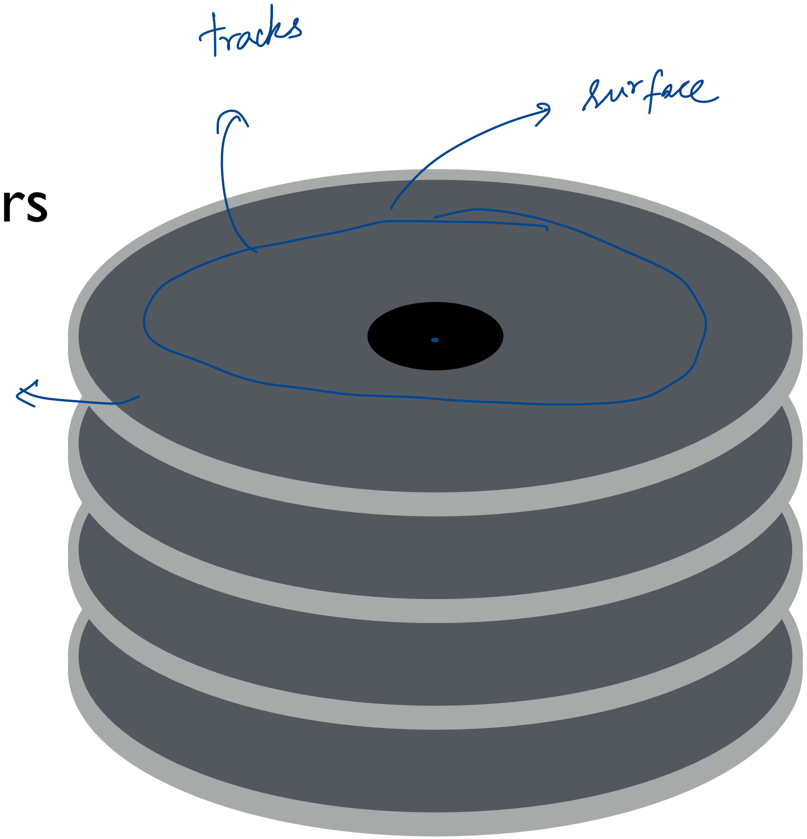
RPM

Motor connected to spindle spins platters

Rate of rotation: RPM

10000 RPM → single rotation is 6 ms

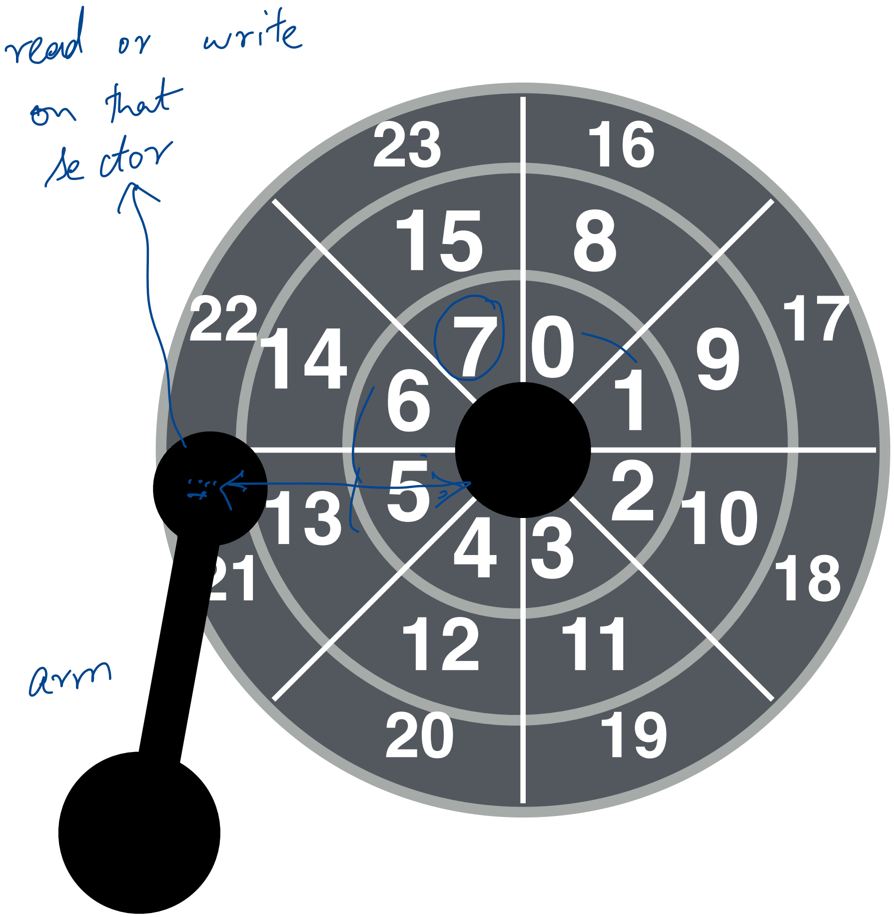
rotations per minute



Heads on a moving arm can read from each surface.

- seek to get to the right track

- wait for rotation right sector is below the head



SEEK, ROTATE, TRANSFER

how far does the arm need to move

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 4 - 10 ms

Average seek = 1/3 of max seek

Spec

Textbook derivation

RPM is given by disk manufacturer

Depends on rotations per minute (RPM)

7200 RPM is common, 15000 RPM is high end

Average rotation: Half of time for 1 rotation

On average we wait for half rotation

Pretty fast: depends on RPM and sector density.

100+ MB/s is typical for maximum transfer rate

Total time = seek + rotation + transfer time

$\frac{\text{data size}}{\text{BW}} = \frac{4\text{KB}}{100\ \text{MB/s}}$ = Transfer time

QUIZ 21

https://tinyurl.com/cs537-sp23-quiz21

What is the time for 4KB random read with 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

$= T_{seek} + T_{rotate} + T_{transfer}$

$= 4 + 2 + \frac{4 \cancel{K}B}{125 \text{ M}\cancel{B}/s}$ ×1000

0.032 ms = 32 μs

= 6.032 ms

≃ 6 ms

T_{rotate} : 15,000 rot — 60 s

1 rot — $\frac{60}{15{,}000}$ s

$= \frac{60{,}000}{15{,}000}$ ms

QUIZ 21

https://tinyurl.com/cs537-sp23-quiz21

What is the time for 4KB random read with Barracuda?

$= T_{seek} + T_{rotate} + T_{transf}$

$= 9\text{ ms} + 4.16\text{ ms} + \frac{4\text{ KB}}{105\text{ MB/s}}$

0.038 ms

$= 13.198\text{ ms}$

Smaller

	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

$T_{rotate} = \frac{60 \times 1000}{7200}\text{ ms}$

$= 8.33\text{ ms} \times \frac{1}{2} = 4.16\text{ ms}$

→ Avg rotation

WORKLOAD PERFORMANCE

So…

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

$= \underbrace{T_{seek} + T_{rotate}}_{4-10ms} + T_{transfer}$

$\downarrow$

large sequential

How does the kind of workload affect performance?

Sequential: access sectors in order → 1 seek + rotate → sequentially read data

Random: access sectors arbitrarily → slow

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 → good

Random → worse / poor performance

Sequential read 100MB: what is throughput for each?

$$T_{transfer} = \frac{100 \text{ MB}}{125 \text{ MB/s}} \approx 0.8 \text{ s} = 800 \text{ ms}$$

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

Example:

read sector 13 → First

read sector 24 After

read sector 14 → Second

read sector 15 → Third

⋮

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 = 60 ms for this sequence

10ms 10 ms 10ms 10 ms 10ms 10ms

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

10ms 10 ms

Reordering can improve performance

SSTF (SHORTEST SEEK TIME FIRST)

Strategy always choose request that requires least seek time
(approximate total time with seek time) → next request as the one with least seek time

Greedy algorithm (just looks for best NEXT decision)

How to implement in OS?
→ How do I know the seek time? Sort by sector number

Disadvantages?
→ Starvation : a disk request that is always waiting.

SCAN

TODO: Create example for unfairness

5, 24, 12, 74, 33
88, 6, 54, 27

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

sector number?

Forward pass 0 to 100

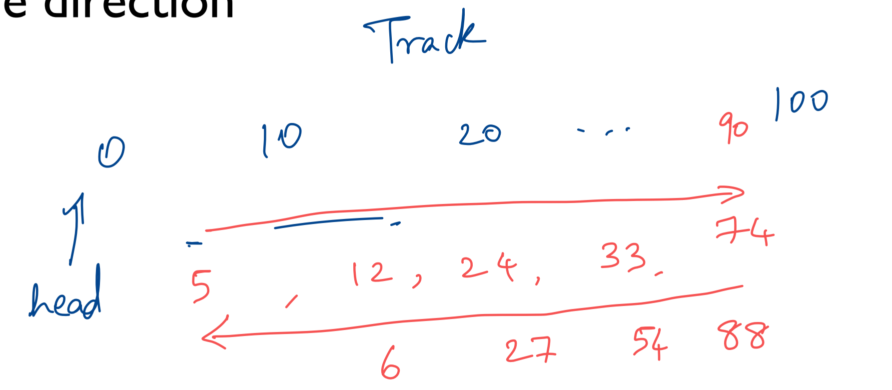
C-SCAN (circular scan): Only sweep in one direction

Avoid fairness across tracks

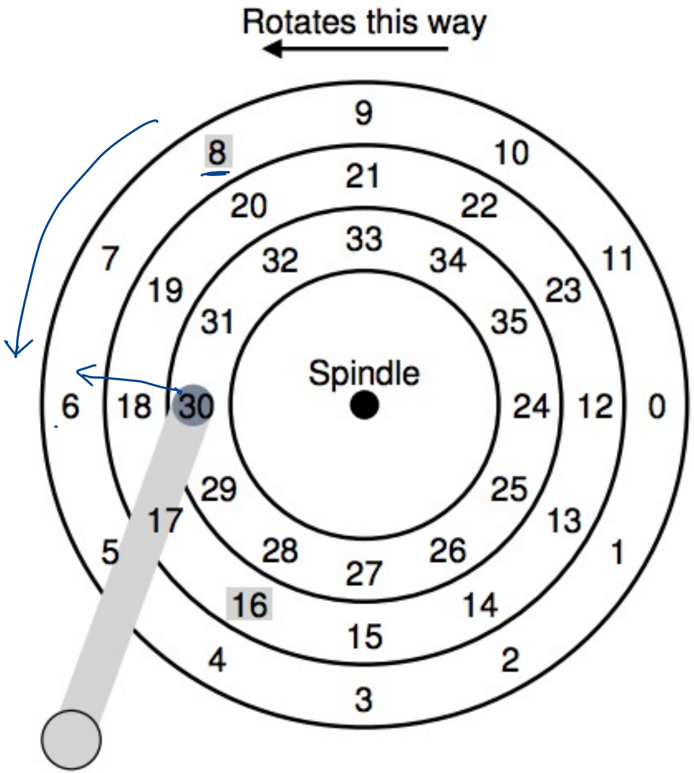
Pros/Cons?

Trade-off

scan can lead to some tracks being read more often



SPTF (SHORTEST POSITIONING TIME FIRST)



More accurate way of doing scheduling

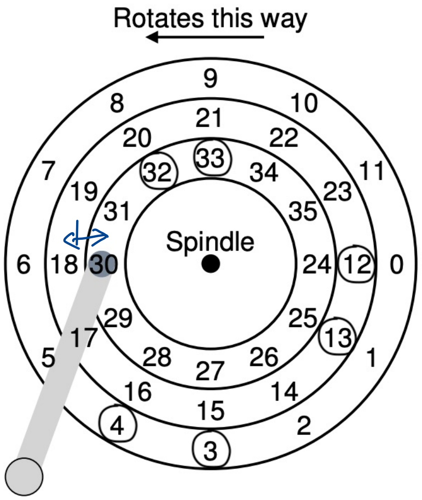
→ need to know lot about disk specification

SATF (SHORTEST ACCESS TIME FIRST)

→ where is the arm right now.

QUIZ 22

https://tinyurl.com/cs537-sp23-quiz22



Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms., 4m for two tracks

What is the time taken when using (FCFS) scheduling?

head is at 30

32 : 2 ms (rotation) = 2 ms

12 : 2 ms (seek) + 2 ms rotate = 4 ms

33 : 2 ms (seek) + 2 ms rotate = 4 ms = 24 ms

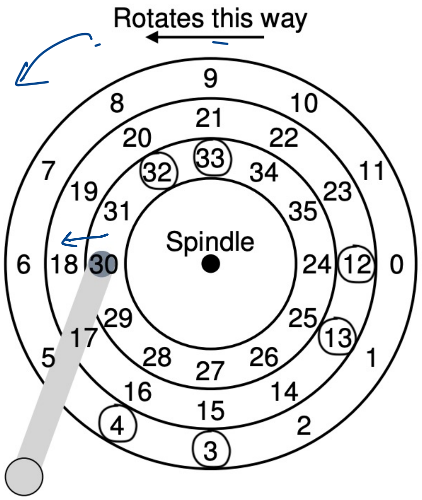
3 : 4 ms (seek) + 2 ms rotate = 6 ms

13 : 2 ms + 2 ms = 4 ms

4 : 2 ms + 2 ms = 4 ms

QUIZ 22

https://tinyurl.com/cs537-sp23-quiz22



Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms.

Order in which requests will be serviced for Shortest Seek Time First (SSTF)?

32, 33, 12, 13, 3, 4

Time Taken :

32 : 2 ms
33 : 0 ms
12 : 2 + 2 = 4 ms = 10 ms
13 : 0 ms
3 : 4 ms
4 : 0 ms

SCHEDULERS

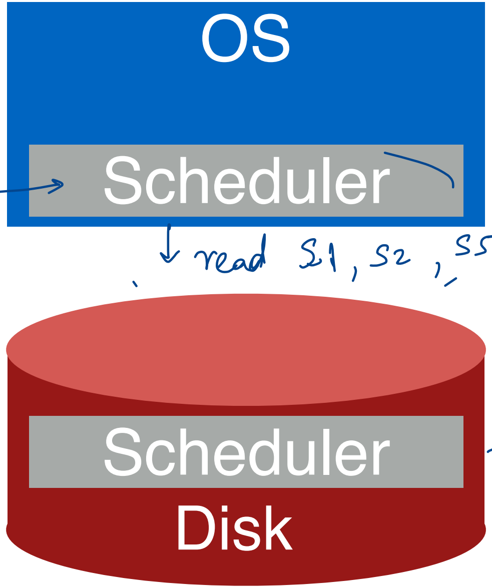
P1 read S1

P2 read S5

OS

- fairness across process,
- multiple policies

upgrade / change



read S1, S2, S5 .. S3

Where should the scheduler go?

knows a lot more

→ density, time taken

→ where is the arm

WHAT HAPPENS?

→ How long do you wait to reorder requests ?

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

File 1

P2: File 2

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

→ 2 sectors

read ahead

P1: Read 300
Read 301

P2: Read 700
Read 701

- - - - - - - -
1ms later

P1: Read 302
Read 303

WORK CONSERVATION

never wait or keep disk idle

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

timeout that you wait

Possible improvements from I/O Merging

Contiguous requests they get merged,

read ahead

SUMMARY

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

I/O Time: rotation_time + seek_time + transfer_time

Sequential throughput vs. random throughput

Scheduling approaches: SSTF, SCAN, C-SCAN

Benefits of violating work conservation

NEXT STEPS

Next class: How to achieve resilience against disk errors