Operating System (OS)
CS232

Persistence: Hard disk drives

Dr. Muhammad Mobeen Movania

Outlines

- What is a HDD
- Physical geometry of HDD
- HDD working
- I/O time and disk scheduling
- Disk head positioning algorithms
- Disk scheduling issues
- Summary

Hard Disk Drive (HDD)

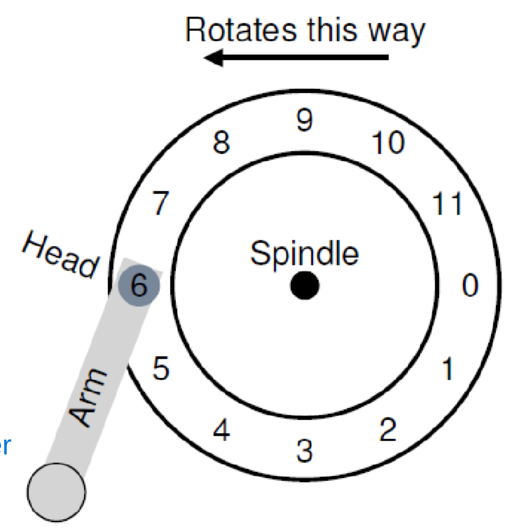
- An I/O device
- Used for persistent storage

  - main form persistent storage for decades
  - file system technology predicated on their behavior
- Consists of *sectors* (512 byte each) numbered *0* to *n-1*
- Writes to a sector are atomic

  - many file systems will read or write 4KB at a time
  - but only write to a sector is guaranteed to be atomic
- Bigger read/writes are possible but not atomic
- Accessing blocks near to each other is faster than accessing blocks far away from each other

HDD – physical geometry

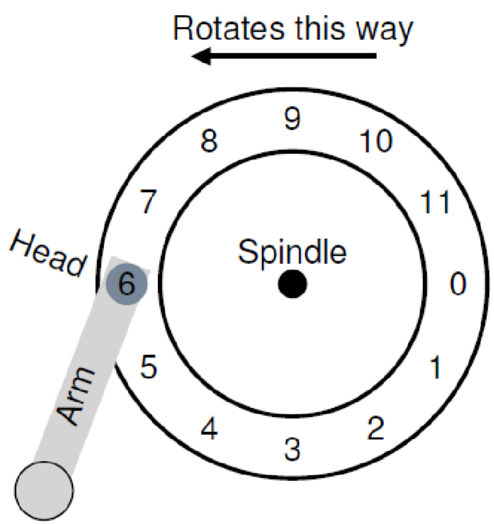
- Platter
  - Made of metal (aluminum) coated with magnetic material on both sides
  - Data is written by inducing magnetic changes
  - A disk may have one or more platters
- Surface: a side of a platter
- Spindle: binds together multiple platters
- Speed: Revolutions per minute (RPM) - 7.2-15k
- Tracks: concentric circle made up of sectors
  - 1000s on a single surface
  - tightly packed (100s fit into width of a hair)
- A Disk Head: is used to read/write data
  - one head per surface
- A Disk Arm: is used to move the head, position it over the right track



HDD – working

- building a model one track at a time
- assume a simple disk with a single track

- Rotational Delay
  - To read track 0?
  - Max rotational delay?

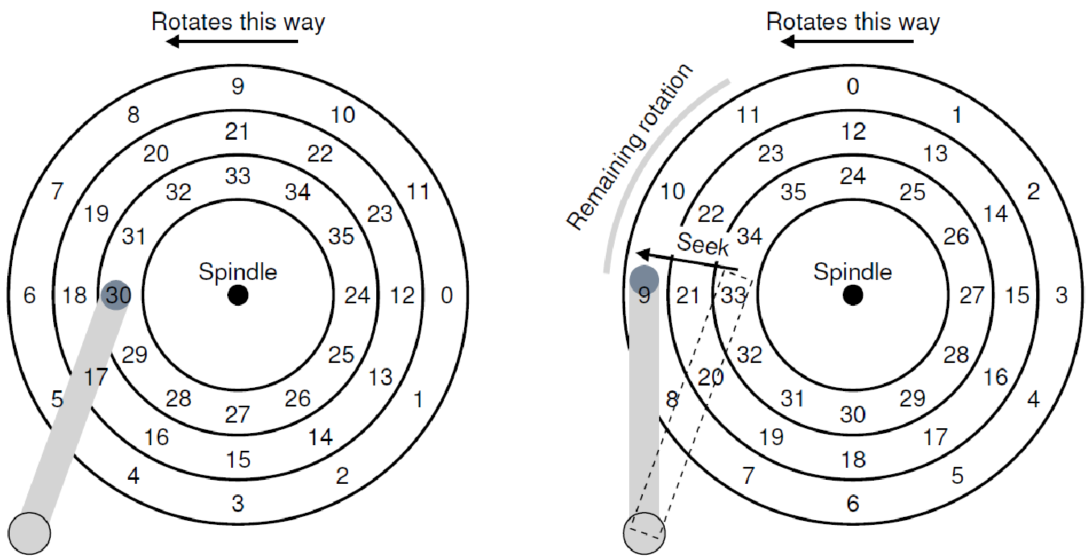


- if R is delay of full rotation
- R/2 to wait for 0
- worst case is sector 5

HDD – working: seek operation

- Seek phases = acceleration + coasting + deceleration + settling

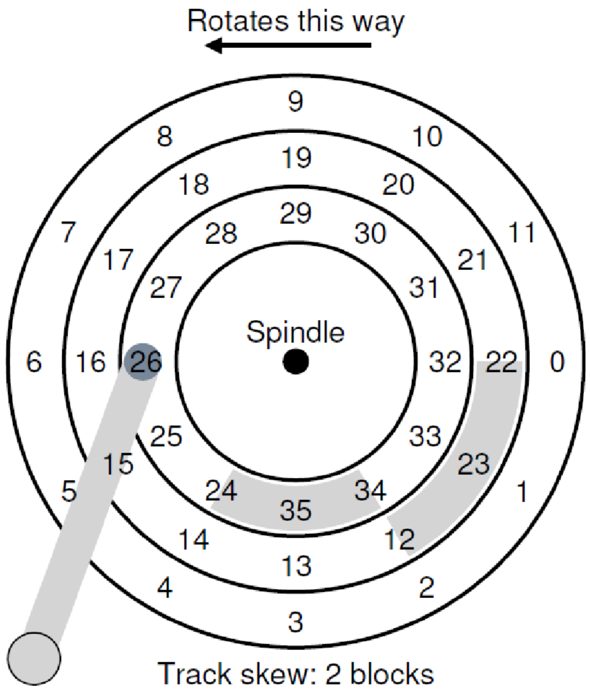
-significant (0.5-2ms) as the drive must be certain to find the right track



- I/O time = seek time + rotate time + transfer

Track skew

- To facilitate transfers crossing track boundaries



Track skew: 2 blocks

- to prevent one full rotational delay when head moves to neighboring track

Track buffer

- Works as a Cache
  - Reads and stores multiple consecutive sectors for future use
- Writing can have two strategies:
  - Write back - acknowledging write completed when written in cache
  - Write through -acknowledging write completed when actually written to disk

HDD specs

	high performance SCSI drive Cheetah 15K.5	built for capacity Barracuda
Capacity	300 GB	slow but pack 1 TB as many bytes as possible
RPM	engineered to spin as fast as possible 15,000	7,200
Average Seek	low seek time 4 ms	9 ms
Max Transfer	transfer data quickly 125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

I/O Time

for a "random" load of 4 KB

- small reads to random locations on
- the disk common in database mgmt

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

$$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$

On the Cheetah:

this is avg seek time.
full seek will be ~2-3 times longer

0.5 * (60 * 1000 / 1500)

4 KB / 125 MB/s

$$T_{seek} = 4\ ms,\ T_{rotation} = 2\ ms,\ T_{transfer} = 30\ microsecs$$ vanishingly small

0.004 MB / 6e-3 s

T I/O = 6 ms, R I/O = 0.66 MB/sec

For barracuda: T I/O = 13.2 ms, R I/O = 0.31 MB/sec

9 + 0.5*(60*1000/7200) + (4000/105000000)*1000

0.004/13.2e-3

Sequential Transfer (100MB)

reads large #sectors consecutively from the disk w/o jumping around

100MB / 125MB/s

- Cheeta T I/O = 800 ms
- Barracuda T I/O = 950 ms

100MB / 105MB/s

	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

- Conclusion: Drive performance varies heavily whether we are doing random small transfers or big sequential transfers !!

Disk scheduling

Module of OS that decides which I/O request to the disk should be scheduled next

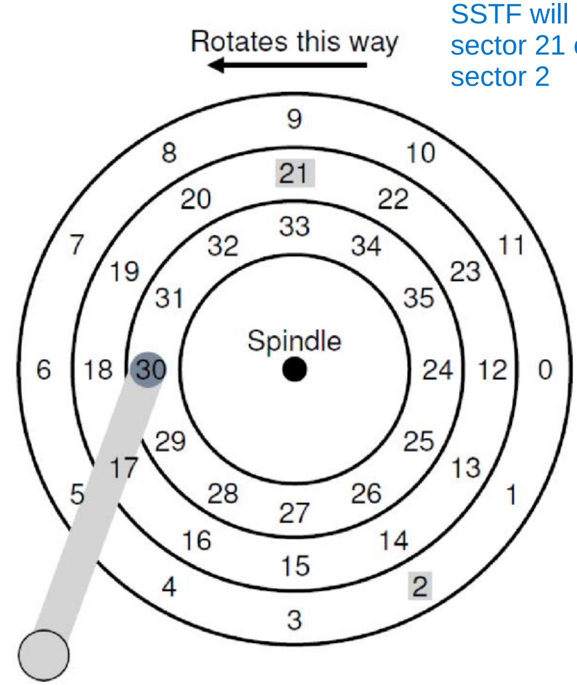
- Multiple I/O requests of varying characteristics
- How can we perform them efficiently?
- Disk scheduler tries to schedule them to reduce average delay.
- Length of job is known ??
- SJF == SSTF == NBF

shortest job first shortest seek time first nearest block first

- when drive geometry is not available to OS
- OS instead sees an array of blocks

- Starvation?

SSTF will schedule sector 21 over sector 2



SSTF can lead to starvation

- a steady stream of requests to the inner track
- requests to any other tracks would be ignored

- SCAN algorithm remedies this
- It moves the head in a *sweep* (from inner to outer tracks and vice versa)
- The next request served will be the closest one but not on the same track!

- freezes the queue to be serviced when it is doing a sweep

- circular scan. sweep in one direction only, then reset

- Variations: FSCAN, CSCAN
- Also called the Elevator algorithm.

- imagine when going floor 10 -> 1, someone got in at 3 and pressed 4 (closer than 1)

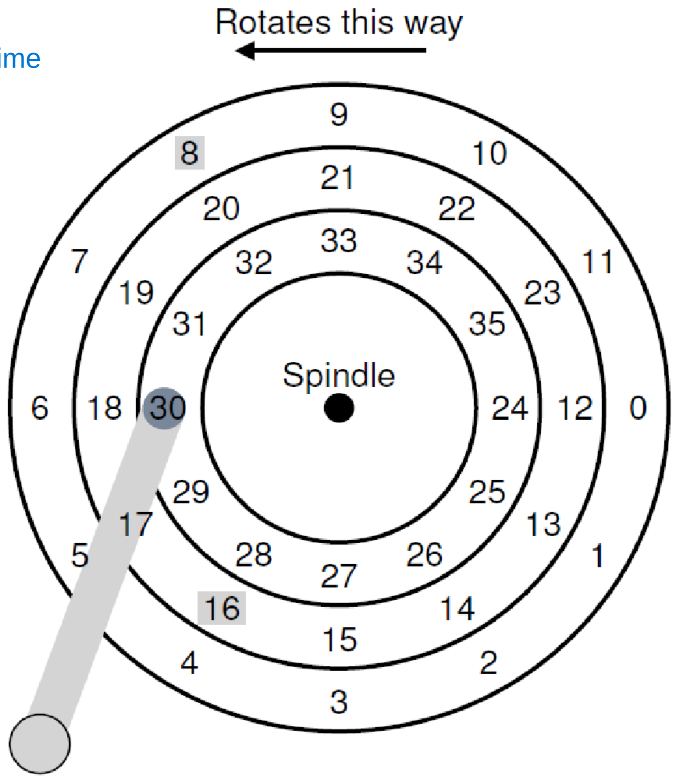
SPTF (shortest positioning time first)

- Which sector to read next: 16 or 8?

- SSTF fine if seek-time >>> rotation time

- SSTF and SCAN do not take into account the time to rotate.
- Problem: often OS does not know the internal details of the disk!

- so it's usually performed inside a drive



Disk Scheduling – issues

- Where is the scheduling performed?
  - OS - happened in older systems
  - HDD - modern systems can implement SPTF accurately
- I/O merging
  - Imagine 3 requests to read from sectors 33, 8, 34 respectively
    - scheduler should merge 33 and 34 into a single request
    - further reordering performed on the merged request
    - important to do at OS level, reduces number of requests sent to the disk
- How long the system should wait before issuing an I/O request to disk?
  - Immediately (work conserving), or
  - After some time (non work conserving) - by waiting a new and "better" request may arrive at the disk, and thus overall efficiency is increased

Summary

- We saw how disk drives work
- We saw different strategies that are used to track the head positioning on tracks and sectors
- OS and disk drive manufacturers try to design a tradeoff for an affordable and performant persistent disk storage