Data-Intensive Computing Systems

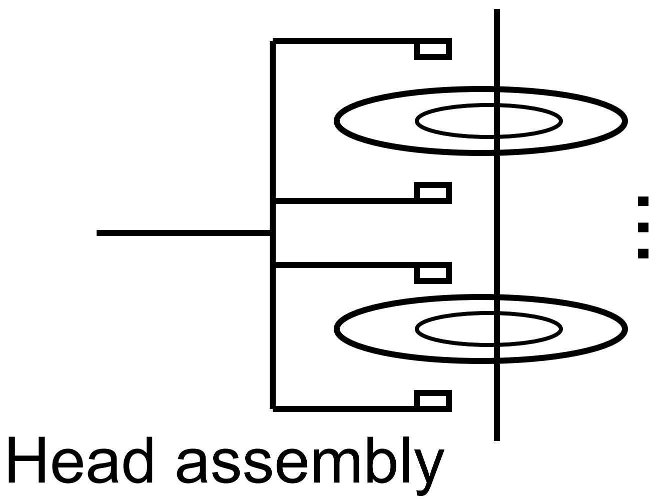
Data Access from Disks

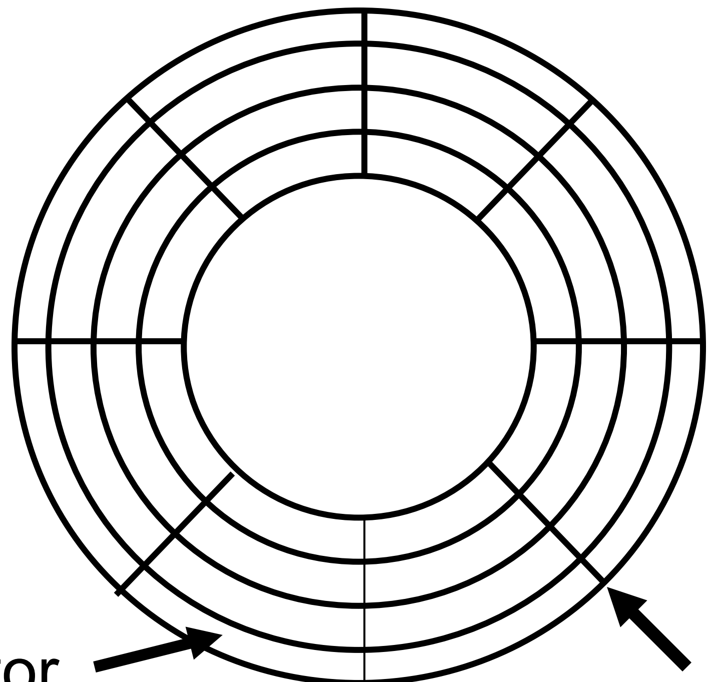
Shivnath Babu

Outline

- Disks
- Data access from disks
- Software-based optimizations
  - Prefetching blocks
  - Choosing the right block size

Focus on: "Typical Disk"



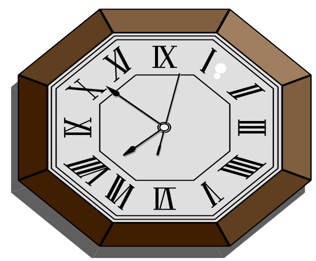


Terms: Platter, Head, Cylinder, Track
Sector (physical), Block (logical), Gap

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Start sector #

Disk Access Time (Latency)

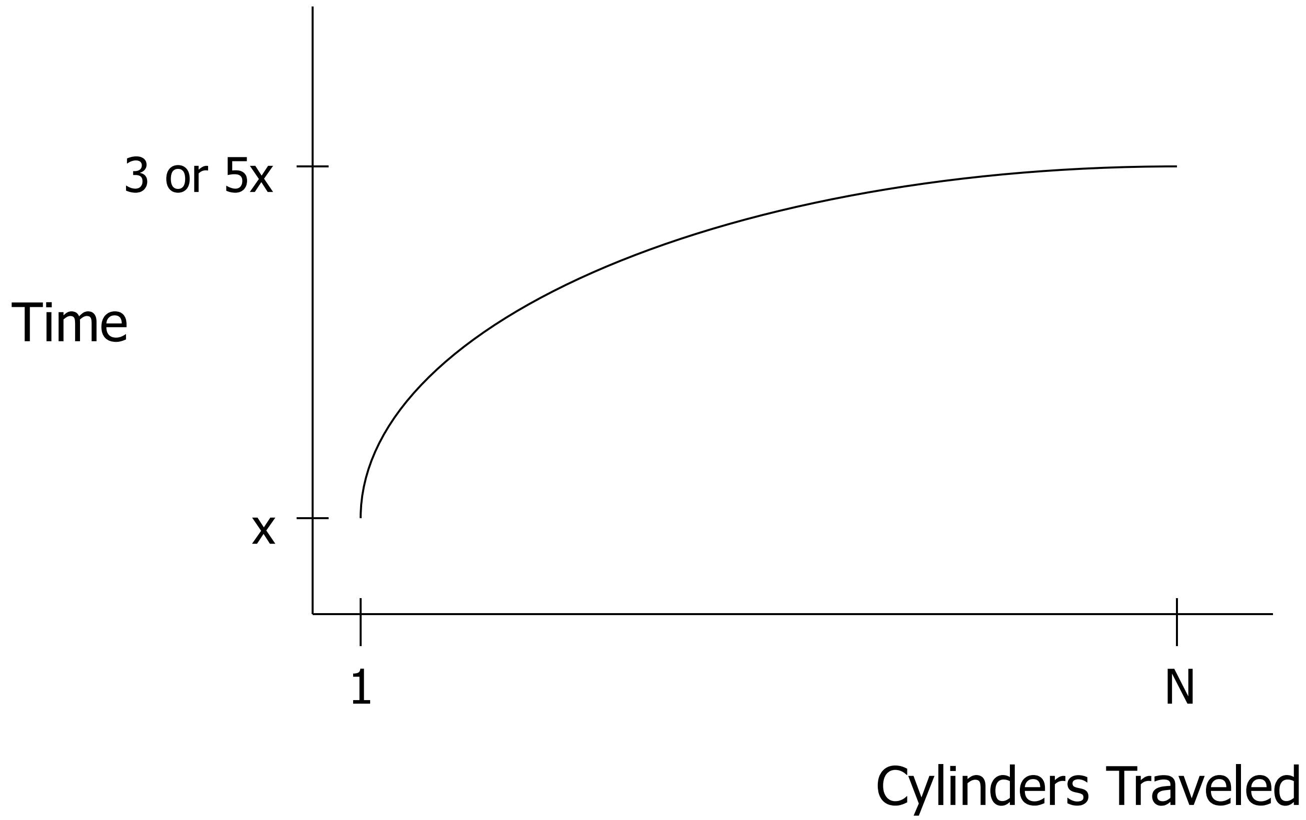
I want block X →  → block X in memory

?

Access Time =

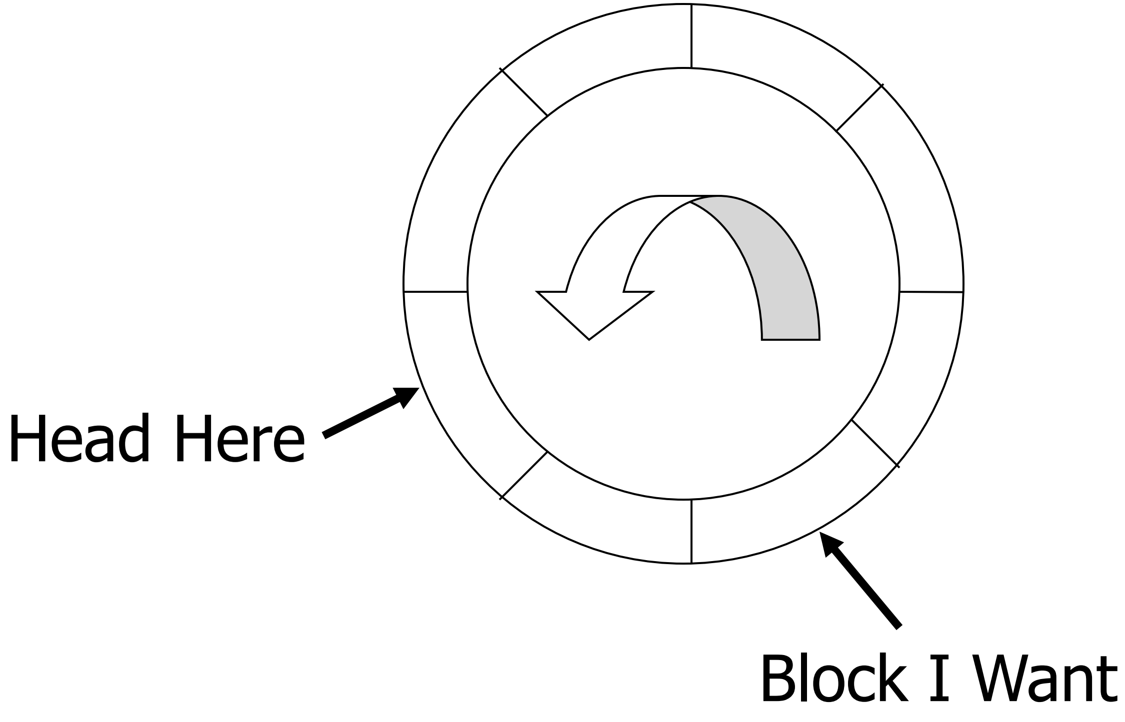
Seek Time +
Rotational Delay +
Transfer Time +
Other

Seek Time



Average value: 10 ms → 40 ms

Rotational Delay



Average Rotational Delay

R = 1/2 revolution

Example: R = 8.33 ms (3600 RPM)

Transfer Rate: t

- t: 1 → 100 MB/second
- transfer time: $\frac{\text{block size}}{t}$

Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

"Typical" Value: 0

- So far: Random Block Access
- What about: Reading “Next” block?

If we do things right …

$$\text{Time to get next block} = \frac{\text{Block Size}}{t} + \text{Negligible}$$

- skip gap
- switch track
- once in a while, next cylinder

Rule of Thumb	Random I/O: Expensive Sequential I/O: Much less

- Ex: 1 KB Block
  - » Random I/O: ~ 20 ms.
  - » Sequential I/O: ~ 1 ms.

Cost for Writing similar to Reading

…. unless we want to verify!

To Modify Block:

(a) Read Block

(b) Modify in Memory

(c) Write Block

[(d) Verify?]

A Synthetic Example

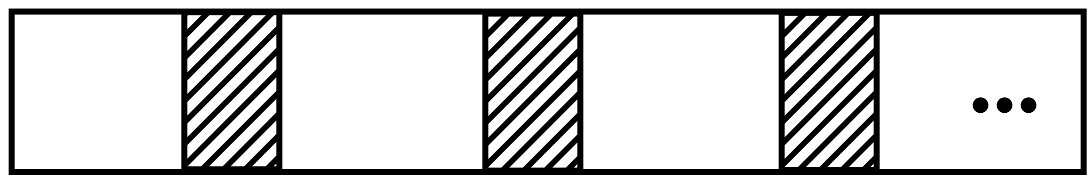
- 3.5 in diameter disk
- 3600 RPM
- 1 surface
- 16 MB usable capacity (16 X 2^{20})
- 128 cylinders
- seek time: average = 25 ms.
  adjacent cylinders = 5 ms.

- 1 KB blocks = sectors
- 10% overhead between sectors
- capacity = 16 MB = $(2^{20})16 = 2^{24}$ bytes
- # cylinders = $128 = 2^{7}$
- bytes/cyl = $2^{24}/2^{7} = 2^{17}$ = 128 KB
- blocks/cyl = 128 KB / 1 KB = 128

3600 RPM → 60 revolutions / sec

→ 1 rev. = 16.66 msec.

One track:



Time over useful data:(16.66)(0.9)=14.99 ms.

Time over gaps: (16.66)(0.1) = 1.66 ms.

Transfer time 1 block = 14.99/128=0.117 ms.

Trans. time 1 block+gap=16.66/128=0.13ms.

Burst Bandwith

1 KB in 0.117 ms.

BB = 1/0.117 = 8.54 KB/ms.

or

BB =8.54KB/ms x 1000 ms/1sec x 1MB/1024KB
= 8540/1024 = 8.33 MB/sec

Sustained bandwith (over track)

128 KB in 16.66 ms.

SB = 128/16.66 = 7.68 KB/ms

or

SB = 7.68 x 1000/1024 = 7.50 MB/sec.

T_1 = Time to read one random block

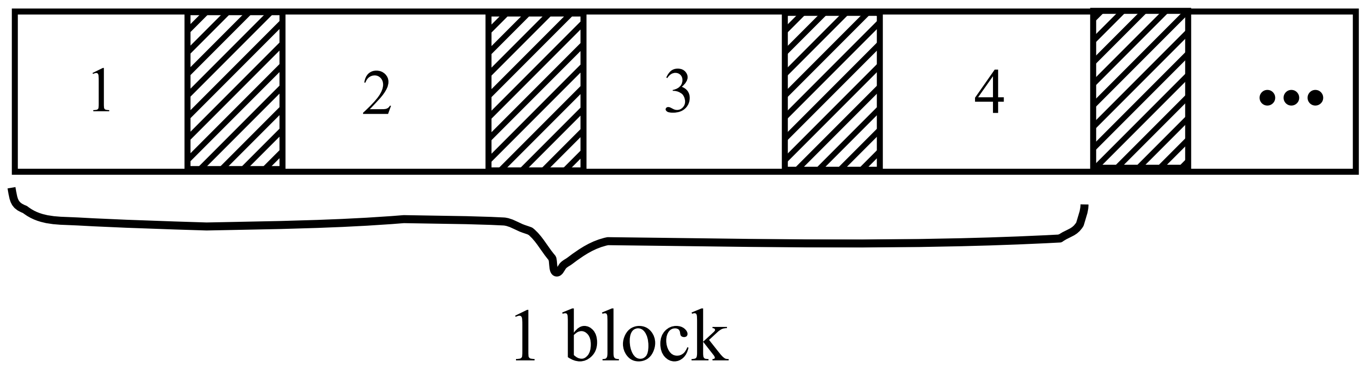
T_1 = seek + rotational delay + TT

$$= 25 + (16.66/2) + .117 = 33.45 \text{ ms.}$$

A Back of Envelope Calculation

- Suppose it takes 25 ms to read one 1 KB block
- 10 tuples of size 100 bytes each fit in 1 block
- How much time will it take to read a table containing 1 Million records (say, Amazon's customer database)?

Suppose DBMS deals with 4 KB blocks



$$T_4 = 25 + (16.66/2) + (.117) \times 1 + (.130) \times 3 = 33.83 \text{ ms}$$

[Compare to T_1 = 33.45 ms]

T_T = Time to read a full track

(start at any block)

$T_T = 25 + (0.130/2) + 16.66^{*} = 41.73$ ms

↑ to get to first block

* Actually, a bit less; do not have to read last gap.

Outline

- Disks
- Data access from disks
- Software-based optimizations
  - Prefetching blocks
  - Choosing the right block size

Software-based Optimizations (in Disk controller, OS, or DBMS Buffer Manager)

- Prefetching blocks
- Choosing the right block size
- Some others covered in Garcia-Molina et al. book

Prefetching Blocks

- Exploits locality of access
  - Ex: relation scan
- Improves performance by hiding access latency
- Needs extra buffer space
  - Double buffering

Block Size Selection?

- Big Block $\rightarrow$ Amortize I/O Cost

Unfortunately...

- Big Block $\Rightarrow$ Read in more useless stuff!

Tradeoffs in Choosing Block Size

- Small relations?
- Update-heavy workload?
- Difficult to use blocks larger than track
- Multiple block sizes