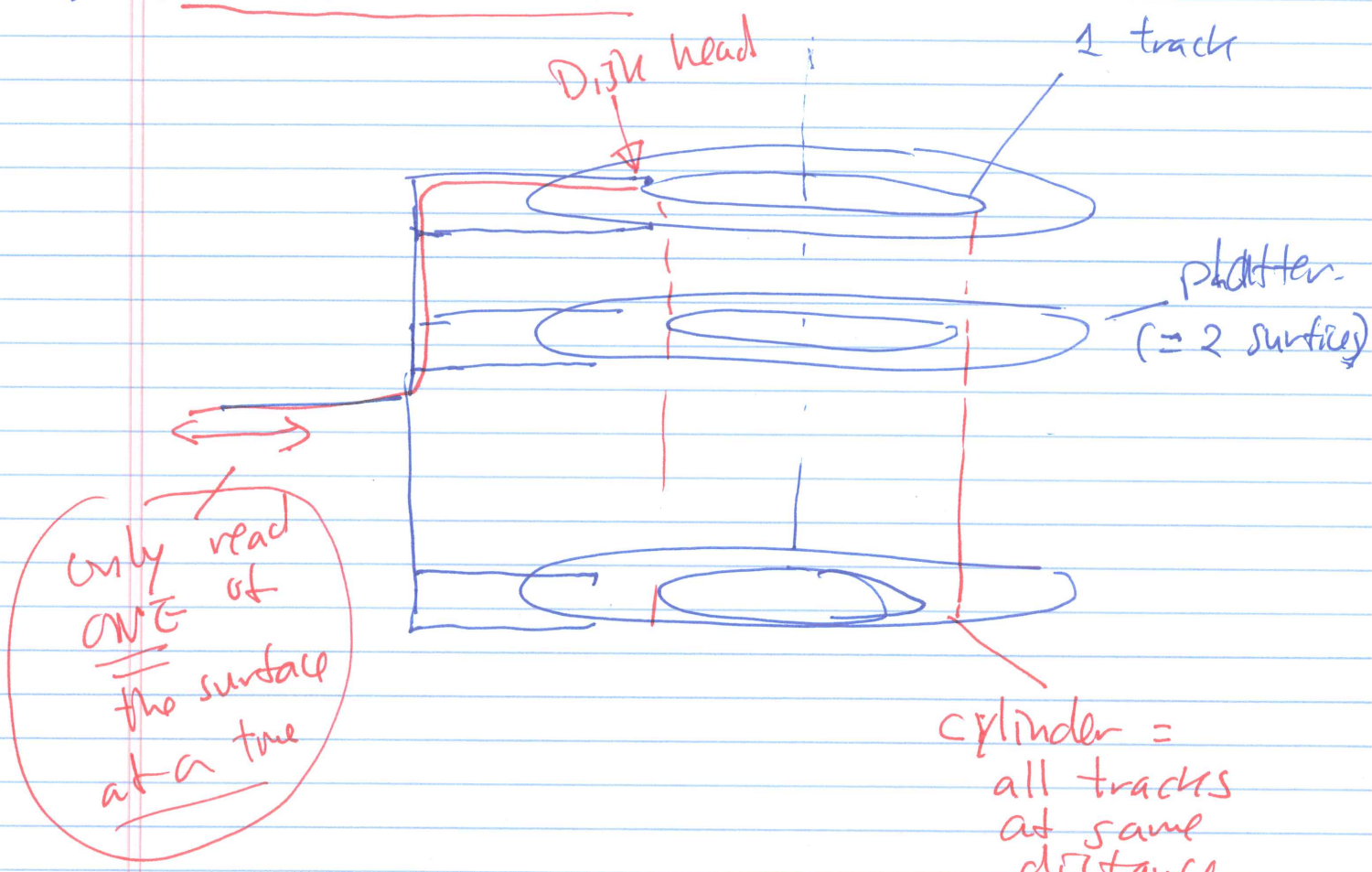
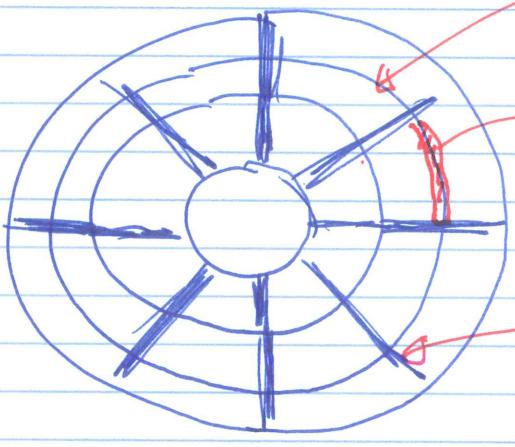
Accessing files on disks

- Structure of a disk:



- Structure of a Disk Surface



2009: ~100,000 tracks per inch

- Blocks, and Directory and files.

Block = logical mapping of sectors into larger (more efficient) units to store files.

eg:

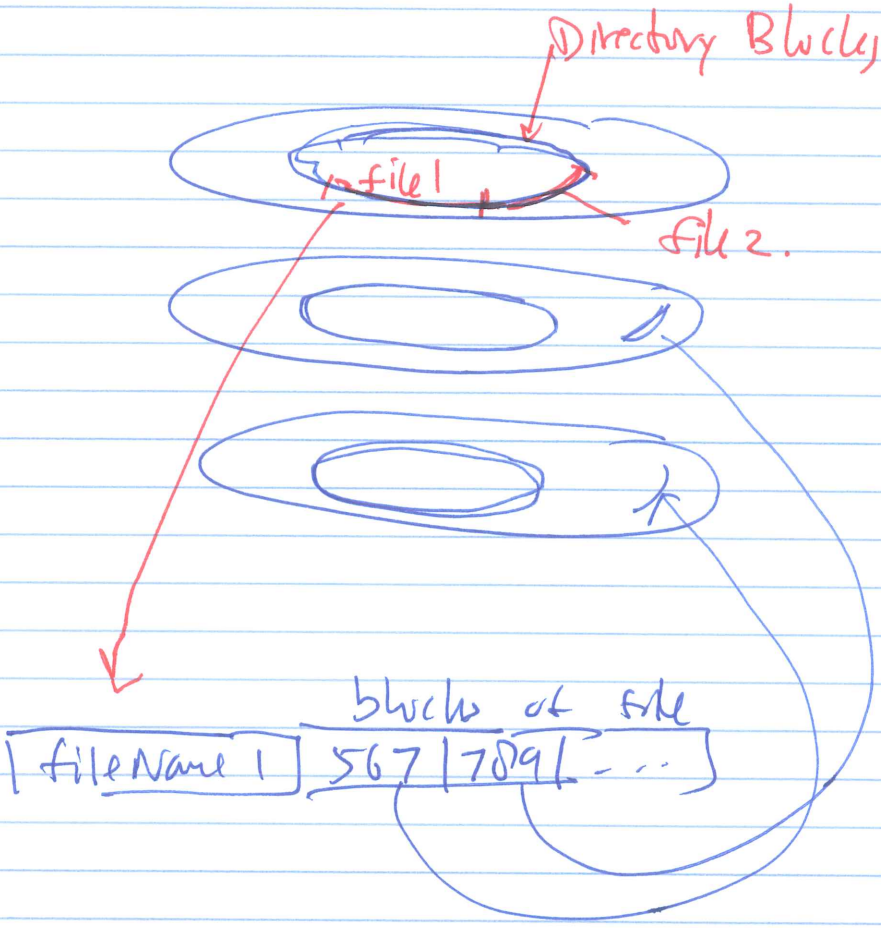
block #	sector #
0	0 1 2 3
1	4 5 6 7
2	8 9 10 11 . .

Directory Blocks = Blocks reserved for storing directory entries

Directory entries contains
(1) file name
(2) block numbers of the file
(3) other info (eg. owner, permission bits etc).

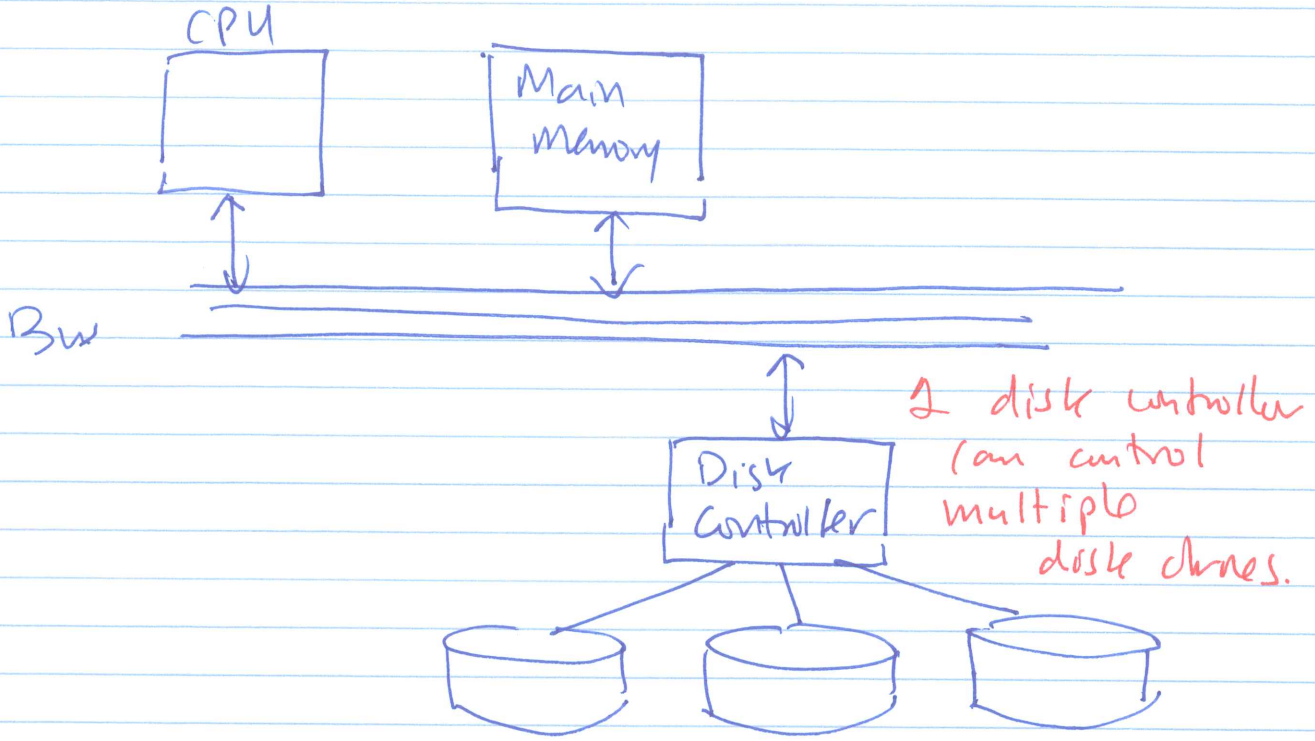
Directory = all directory entries stored in the Directory Blocks
(contains data about the files).

Example:


Directory Blocks.
file1
file2.
blocks of file
fileName 1
567
789
...

The Disk Controller

Connecting Disks to a computer system:



Disk controller: small (simple) processor that:

(1) control movement of the disk heads (move head to a specific track).

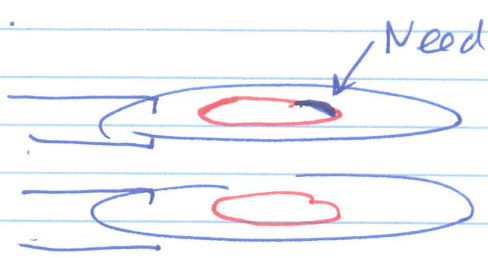
(2) select a sector on a track.

(3) transfer data (bits) between disk surface and computer main memory.

(4) Has an internal buffer to speed up data transfer (can buffer a track or more).

Disk Access Time.

Start:



Steps the disk takes to access the data on a specific sector:

(1) move the disk head(s) to the track containing the sector. ⇒

⇒ seek time : 0 ~ 10 msec

0: same track

10 msec: one end to another.

avg. seek time ≈ 5 msec.

(2) wait for the sector to come under the disk head.

⇒ rotational latency.

round ≈ 10 msec

: 0 ~ 10 msec also.

avg. rt. latency ≈ 5 msec.

(3) Transfer data. (< 1 msec).

What do you take away from this:

(1) Data processing proceeds piecemeal:
1 block at a time.

(2) 1 block ~ 4 K – 64 K bytes.

(3) Time needed to bring 1 block of data into main memory:

~ (estimate) 50 msec.

(4) In 50 msec = 50.000 μ sec
= 50,000,000 nsec.

A CPU (with good/large cache)
can execute over 10,000,000 instructions

More than enough to process 64 K byte of data!!!

Called the "I/O Model of computation"

=> In DB applications: Disk access time will be the dominant latency! (response time)

Accelerating Accesses to Disks.

- Techniques used to speed up file access:

(1) Place blocks that are accessed (frequently) together on the same cylinder.

⇒ because seek time = 0 !

seek time = $\frac{1}{2}$ the total access time !!!

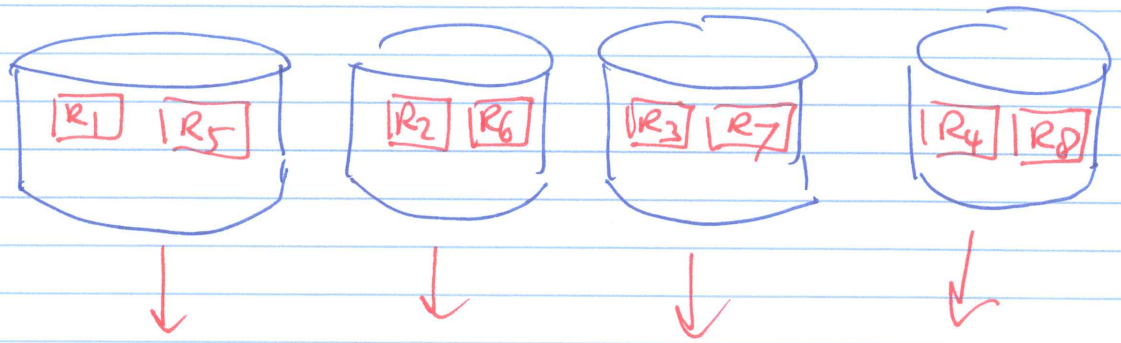
(2) Use multiple smaller disks instead of one large disk.

Multiple disks ⇒ multiple disk heads move independently.

⇒ increase # blocks accessed per sec.

Striping: store disk blocks of a file over many disk drives.

eg:



they can retrieve the data independently.

(Controller can control all disks at the same time).

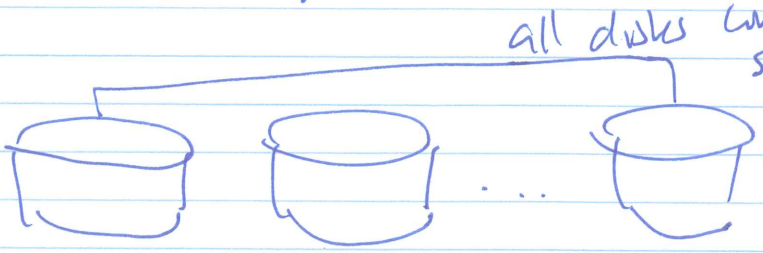
- control signals are short

Experience: with 4 disks, avg. speed up = 3× speed of 1 disk.

(from book).

(Assumption: transfer speed between Controller ⇄ disks is high enough to support n simultaneous transfers.)

(3) Mirror disks. — very expensive ...
— get added reliability



read from any disk

Read access speed = (n) * speed of 1 disk

Both: write speed ≃ same as 1 disk.

Assumption: transfer speed

Controller → disks

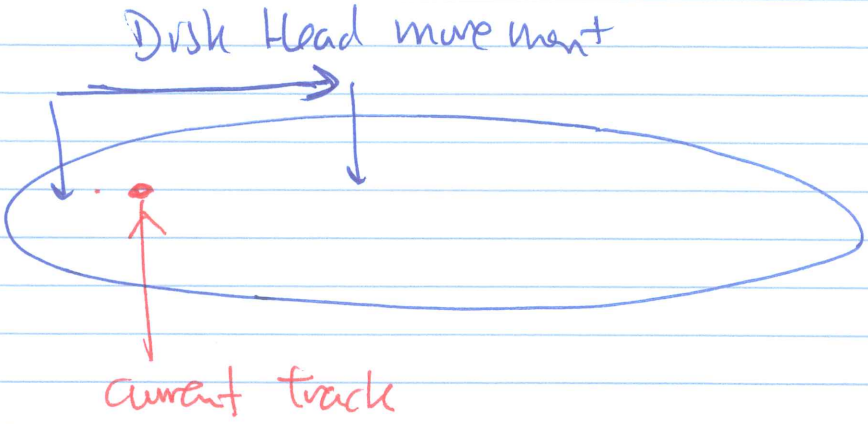
sufficiently large.

④ Use a disk-scheduling algorithm to reduce disk access time.
(the scheduling alg. with re-order the disk requests !!!).

⇒ Commonly used disk scheduling alg:

= The "Elevator" Algorithm.

Elevator Alg:



⇒ Process ALL requests for sectors residing on the current track.

⇒ Move disk head to the next track (same direction) with request.

(Operate like an "elevator" picking up people).

(5) Prefetching blocks

In some applications (eg: sort - read all blocks), we can predict the order in which blocks will be requested.

⇒ load blocks into main memory BEFORE they are requested!

⇒ Best combine with Disk scheduling Algorithm (save seek time !!!)