I/O and disk systems

Note Title

Topics today:
1. Amdahl's Law
2. I/O Architectures
3. Data access and transfer times
4. RAIDS

① Amdahl's Law

Suppose we have a system that processes tasks at a fixed rate. (e.g. a database that processes 2,000 requests per second). Suppose we have the option to speed up some component of the system. (e.g. we could rewrite one particular type of query in the database system).
Suppose

f - is the fraction of time currently consumed by the component.

k - is the factor by which we can speed up the component

Then the speedup factor S from improving the component is

$$S = \frac{1}{(1-f) + f/k} \quad \Leftarrow \text{Amdahl's Law}$$

Example: the queries under consideration occupy 30% of processing time. The rewrite will make them 50% faster. What is resulting speedup?

Solution: do as exercise

See worked example in Section 7.3 of the book for how to do a cost/benefit analysis of two different proposed improvements.

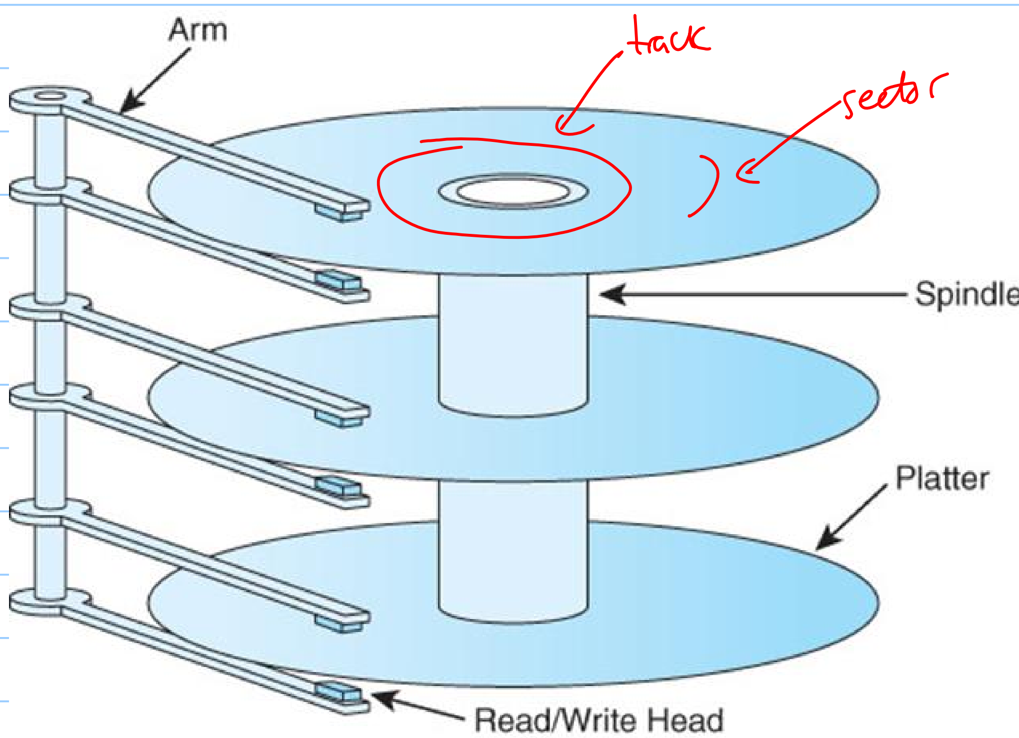
(2) I/O Architectures

Detailed knowledge is not required. We just need an awareness of the following basic facts:

- The most common form of I/O uses interrupts, as described earlier in the semester.
- An I/O device signals an interrupt when ready to send or receive data. CPU suspends regular operation of the fetch-decode-execute cycle and services the interrupt instead.
- Efficiency is greatly improved using direct-memory access (DMA) – a special piece of hardware that transfers data between devices and main memory. The CPU must initiate the transfer, but can then do other useful tasks while the transfer is being performed.

(3) Data access and transfer times

Disk internals, from fig 7.14 of the textbook:



Important definitions:

- Seek time – time for disk arm to move to desired track (typically a few ms).

- Rotational latency – time for desired data to rotate to location under disk head.

e.g. for 7200 RPM disk,

seconds per rotation $= \frac{60}{7200} = 0.0083$

On average wait for $\frac{1}{2}$ a rotation, i.e. 0.004 s.

So rotational latency $\approx$ 4 ms.

- Access time = seek time + rotational latency
  = (for example) 3ms + 4ms
  = 7 ms

- Transfer rate — rate at which sequential data can be transferred. (typically 100 MB/s as of 2010).

- Total time to read some sequential data

$$= \text{access time} + \frac{\text{data size}}{\text{transfer rate}}$$

Note: Caching is used everywhere in the storage heirarchy. e.g.
- the OS keeps a file system cache (100s of MB) in main memory
- the disk caches recently-accessed blocks in its controller

④ RAID 5

RAID ≡ "Redundant Array of Independent Disks"

- an array of disks with a controller that "looks like" one big disk, to the computer.

- employs two main ideas : <u>striping</u> and <u>parity</u>

(A) Striping

Striping means that logically contiguous parts of a file are stored on different disks. The chunks stored on each disk are called <u>stripes</u>. A Typical stripe size is 64kB.

Simple example with 4 disks and stripe size of 3 bytes, storing the string "ABCDE..." :

	disk 0	disk 1	disk 2	disk 3
byte 0	A	D	G	J
1	B	E	H	K
2	C	F	I	L
3	M	P	S	V
4	N	Q	T	W
5	O	R	U	X
6	Y			
7	Z			
8				
⋮	⋮	⋮	⋮	⋮

(B) Parity

parity ≡ XOR ≡ binary addition without carries

Example: given binary data stripes 101, 111, 011, what is their parity?

Solution:

$$\begin{array}{r} 1\,0\,1 \\ 1\,1\,1 \\ 0\,1\,1 \\ \hline 0\,0\,1 \end{array}$$

parity is 001

Important property of parity: if we lose one of the data stripes (e.g. due to a disk failure), we can use the parity stripe to recover it!

e.g.

$$\begin{array}{r} 1\,0\,1 \\ 1\,1\,1 \\ \oplus \quad ?\,?\,? \\ \hline 0\,0\,1 \end{array}$$

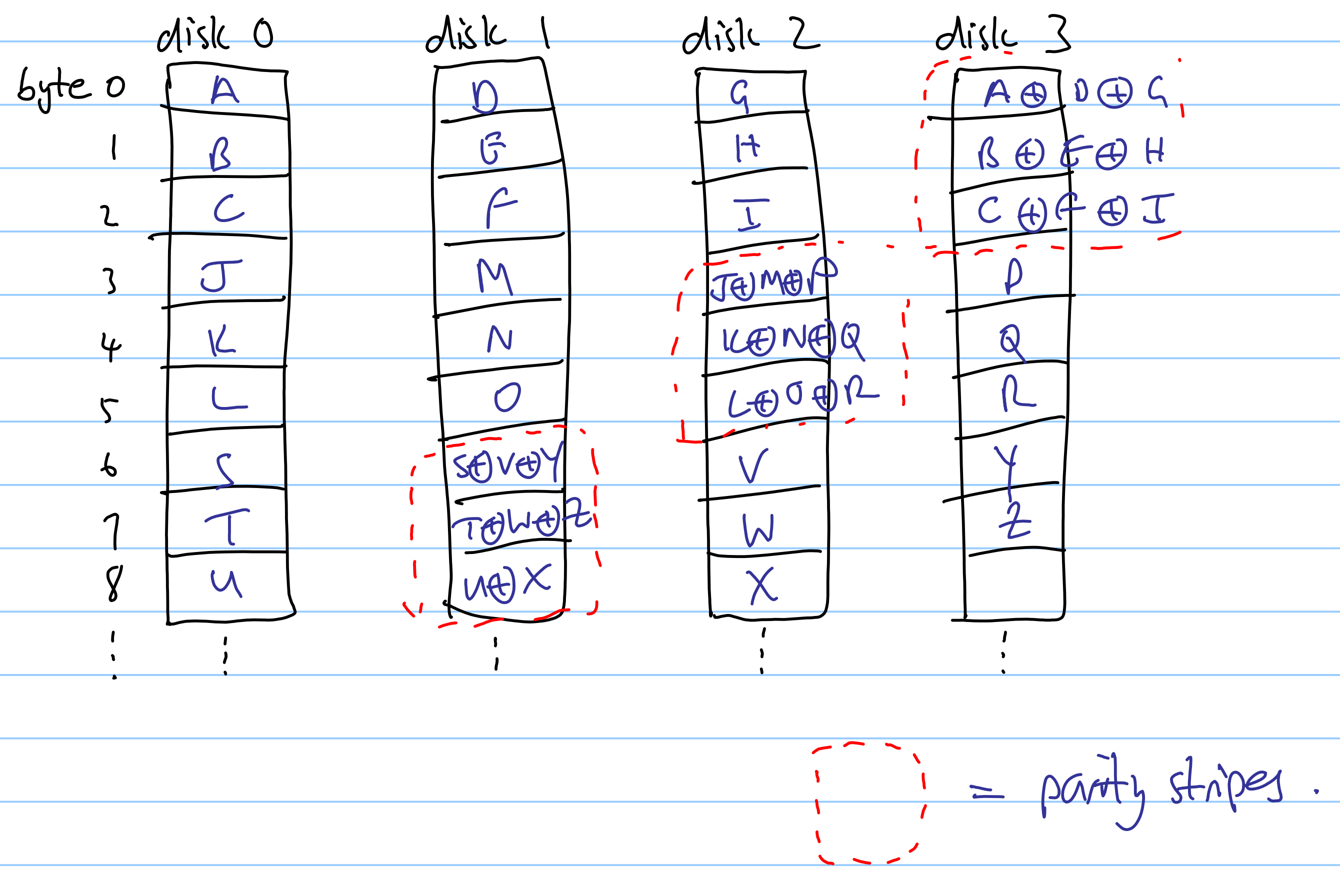
← recover by XORing everything else:

$$\begin{array}{r} 1\,0\,1 \\ 1\,1\,1 \\ \oplus \quad 0\,0\,1 \\ \hline 0\,1\,1 \end{array}$$

← recovered stripe

The RAID5 scheme stores parity stripes to enable recovery from failure of an entire disk.

Example, again storing "ABC..." with stripe size 3 bytes:



What happens if we lose disk 1? Recompute each stripe by XORing the appropriate stripes from the other disks.:

byte	disk 0	~~disk 1~~	disk 2	disk 3
0	A	D	G	$A \oplus D \oplus G$
1	B	E	H	$B \oplus E \oplus H$
2	C	F	I	$C \oplus F \oplus I$
3	J	M	$J \oplus M \oplus P$	P
4	K	N	$K \oplus N \oplus Q$	Q
5	L	O	$L \oplus O \oplus R$	R
6	S	$S \oplus V \oplus Y$	V	Y
7	T	$T \oplus W \oplus Z$	W	Z
8	U	$U \oplus X$	X	
⋮	⋮	⋮	⋮	⋮

e.g. disk 1, byte 0 = $A \oplus G \oplus (A \oplus D \oplus G) = D$

byte 5 = $L \oplus (L \oplus O \oplus R) \oplus R = O$

byte 6 = $S \oplus V \oplus Y = (S \oplus V \oplus Y)$