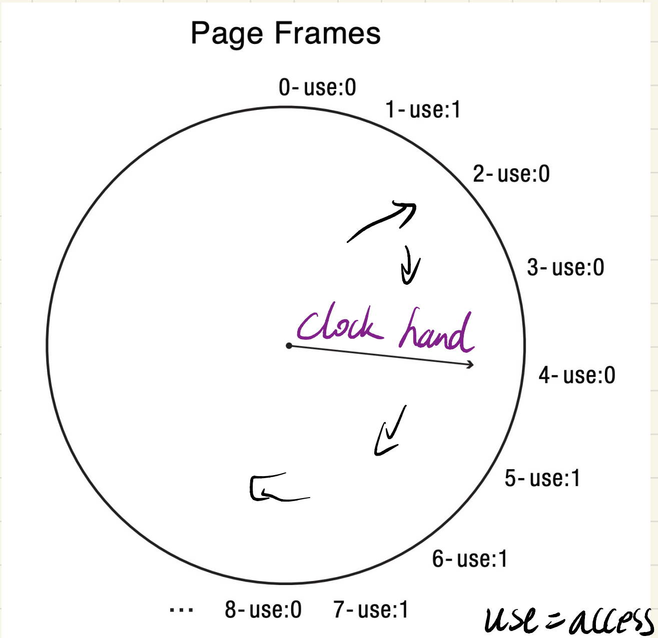
11/13/23

Clock



use = access bit

use/access bit for each page set by hw

-> find less recently used page by looking at access bit
-> access bit is 1 => clear to 0
-> clock hand moves after each run
-> only cares about access bit

when evicting a page that's dirty,
needs to write it to swap

when evicting a page that's clean (dirty bit == 0),
no need to write to swap

↓

code page,
static data page,
unwritten stack

Second Chance /Enhanced Clock

Access Bit	Dirty bit	
0	0	evict the page
0	1 ⇒ 0	clear the dirty bit, track cleared dirty page, move on
1 ⇒ 0	0	clear the access bit, move on
1 ⇒ 0	1	clear the access bit, move on

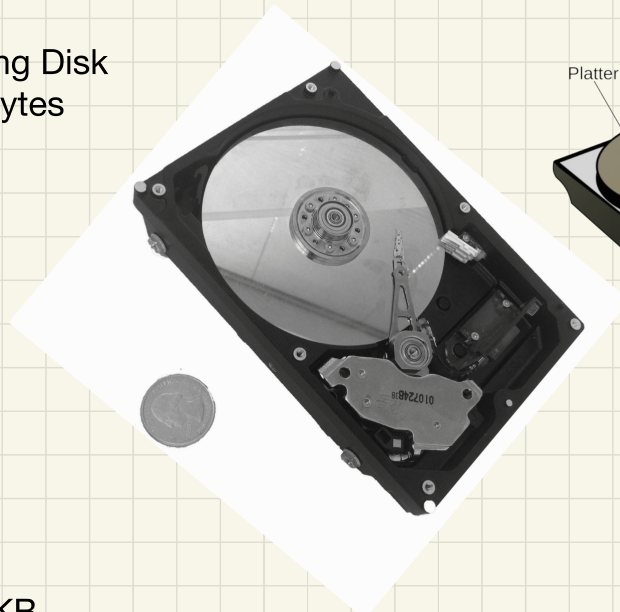
Storage Devices

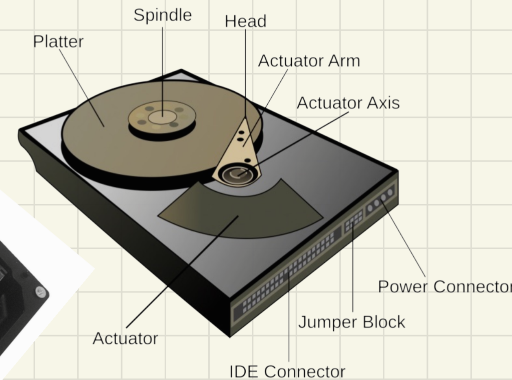
persistent, not byte addressable, block level access

TB in capacity
Much cheaper than DRAM

Hard Disk / Spinning Disk

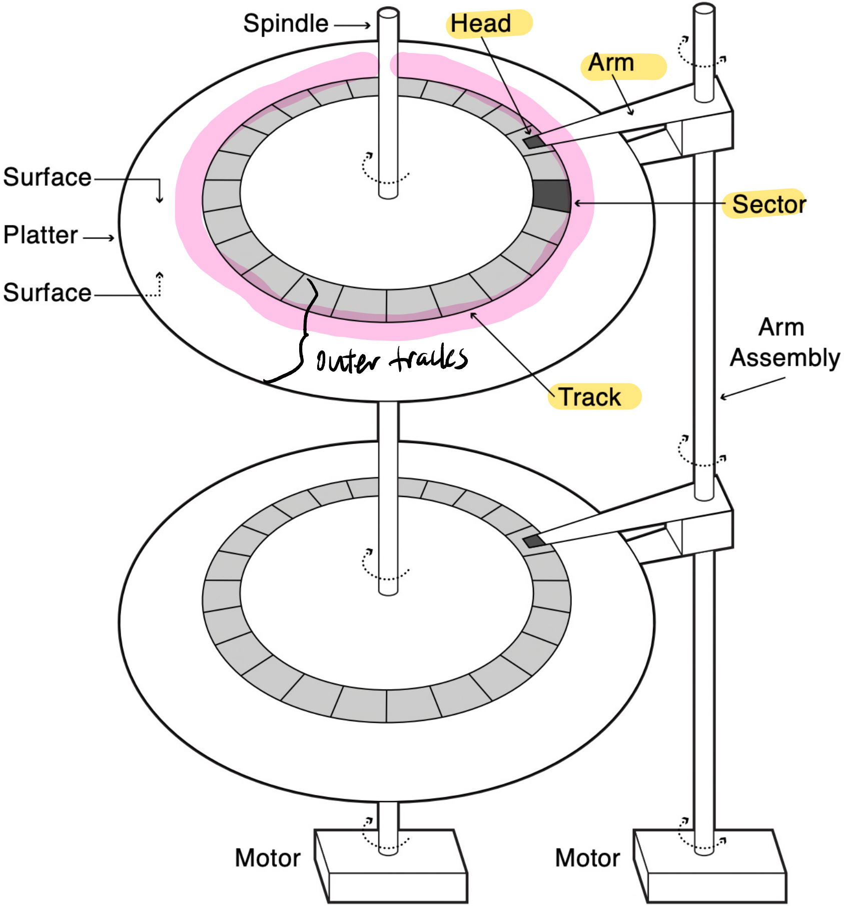
- -> sector: 512 bytes
- -> 10-20$TB





Solid State Drive

- -> block size 4 KB
- -> 3-5x cost of HDD



disk read steps:

kernel sends the request to disk controller (ide.c)

disk finds the right platter & surface moves arm to track containing the sector

waits for desired sector to rotate under the head & reads the sector

data is transferred back to the host

Disk Performance

tota time = seek time + rotation time + transfer time
(arm to track) (sector under disk head) (data transfer)

1). seek time : 1-20ms depending on how far to seek (let's say 10ms on average)

2). Rotation time : specified as RPM, eg. 7200 RPM = 120 RPS = 0.12 RPMS
(assume it takes half a rotation for the desired sector to be in the right place) → need to convert to ms per rotation

$\frac{1}{0.12}$ = 8.3 ms per rotation

average rotation time = half of full rotation ≈ 4 ms

3). Transfer time : specified as disk bandwidth.

Example : read 1 sector, seek time 10ms, 7200 RPM, bandwidth 120 MiB/s

total time = 10ms (seek) + 4ms (rotation) + 0.004 ms (transfer time of 512 bytes)

= 14.004 ms

Reading 10 sectors

→ 10 consecutive reads (sequential) : 1 seek + 1 rotation + transfer time of 5120 bytes

$10 + 4 + 0.04 \text{ ms} = 14.04 \text{ ms}$

→ 10 random reads/writes : 10 seek + 10 rotation + transfer time of 5120 bytes

$14.004 \times 10 = 140.04 \text{ ms}$

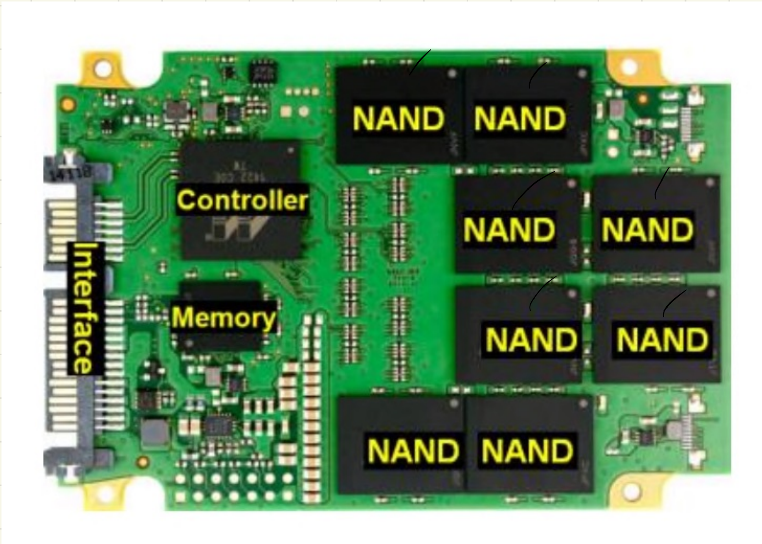
Metrics : IOPS (I/O operations per second)

→ $\frac{\# \text{ of I/O operations}}{\text{total time (s)}}$

→ 10 sequential reads : $\frac{10}{0.01404} = 712$ IOPS

→ 10 random reads = $\frac{10}{0.1404} = 71.2$ IOPS

SSD.



↙ no moving parts
parallel accesses

units: page (2-4KB).

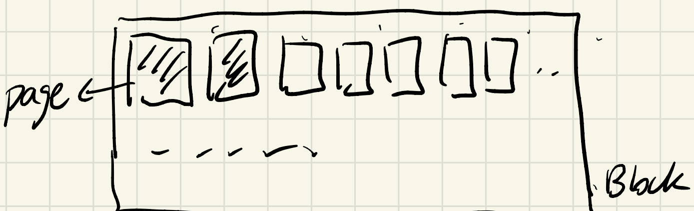
(erasure) block (1-8 MB)

Operations:

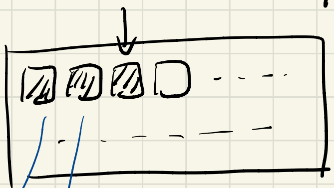
→ read a page

→ erase an entire block
(set all bits to 1s)

→ program a page
(can only program an empty page)
write 0s

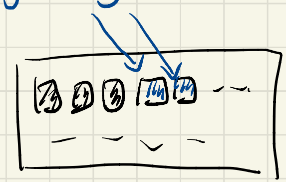


To do inplace update on SSD:

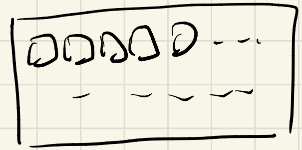


Block

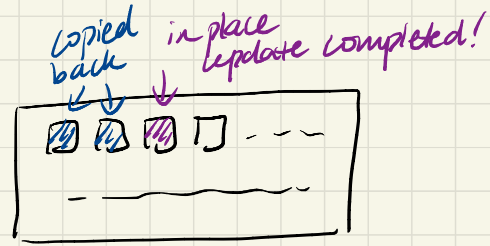
copy existing pages elsewhere



random block w/ empty pages



erase entire block

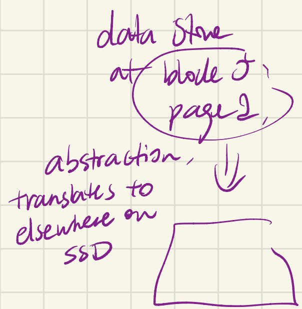


update page w/ new content

☆ doing this causes a large # of writes & repeated writes to updated pages

→ SSD pages have limited write cycles (10-100k)

→ wear leveling = spread writes across blocks/pages to reduce repeated writes to a single page.



Logical address

⇓

actual block/page