Review (537) 5/7

=> Open

General

Suggestions: => focus on now
=> what?
=> How?
Read + Do

1/3 RAID
3/5 FFS : Large file + chunk (math) sizes
3/5 SSDs + FTLs + Hybrid Maps
1/3 Journaling : why good? why bad?
4 Disk Sched: SATF

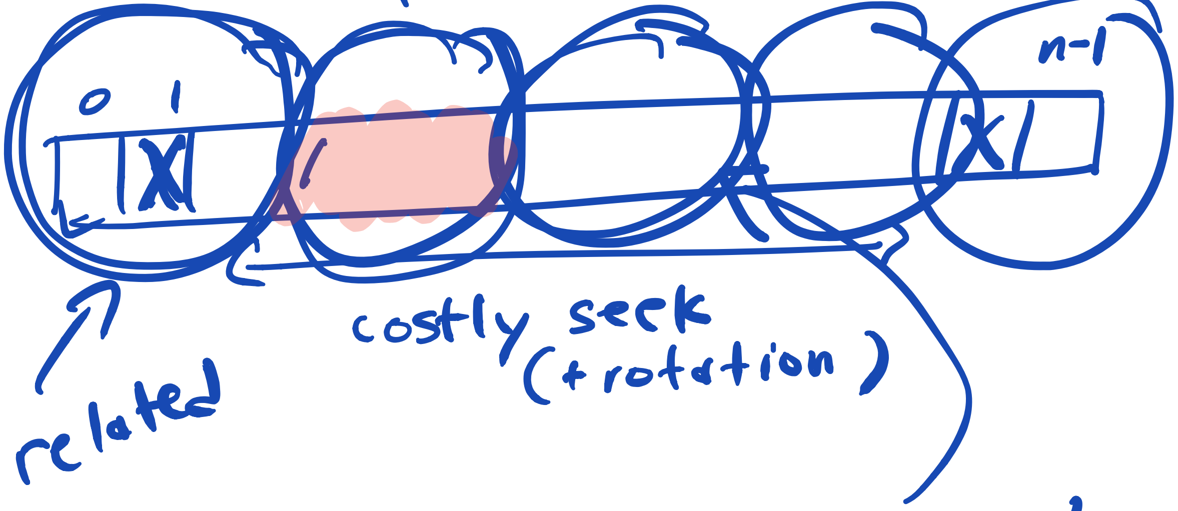
Old Exam Questions

S '16, Q. 7
F '11, Q. 4, 3, 6, 5
S ~~'18:~~ ~~All~~ !

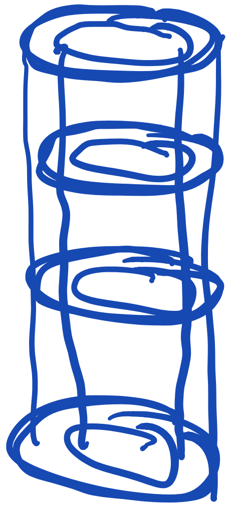
~~[illegible]~~ 6
M + F 2
50%
—
—
7

FFS: Fast File System

=> Groups + Locality
esp. on Hard Drives



=> Cylinder Groups
(Block)



FFS:
place related items
into same group
(spread out
unrelated item)

⇒ related? (heuristic)

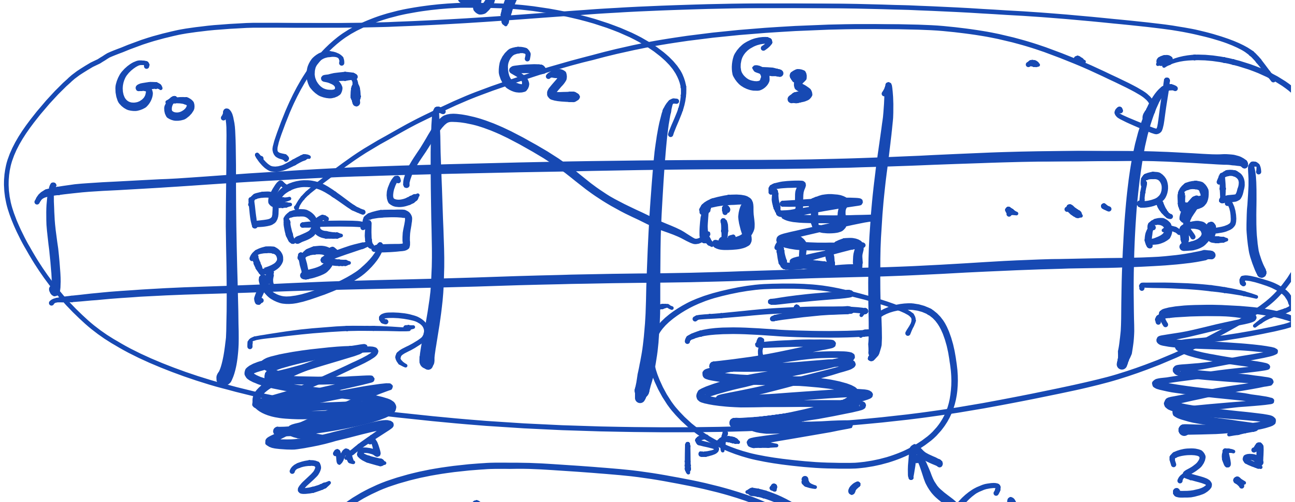
⇒ inode + its data

⇒ files in same directory

(make xv6?)

Exception: Large Files

Avoid over-filling a group w/ one file



cost: performance

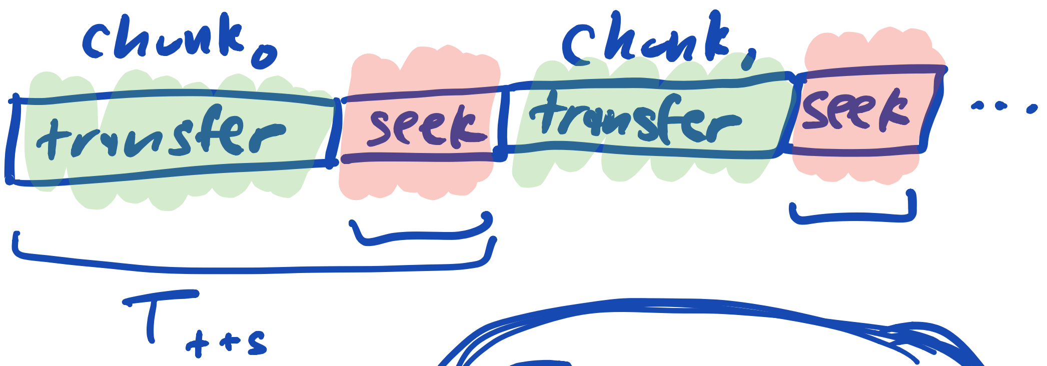
old: one large seq transfer

new: shorter seq xfer, seek (repeat)

⇒ how to get good perf?

A: empirically answer

how to determine chunk size?



make high: $\frac{T_{transfer}}{T_{transfer} + T_{seek}}$

⇒ .9 (90%)

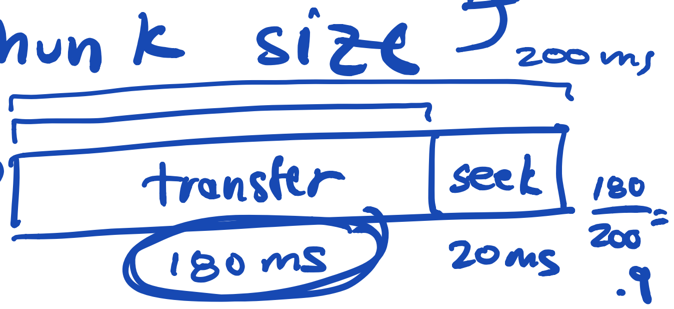
Disk:

⇒ Transfer: 100 MB/s

⇒ Avg seek: 20 ms

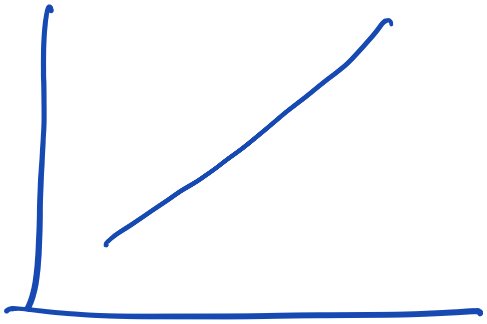
Compute Chunk size

(18 MB) ⇒

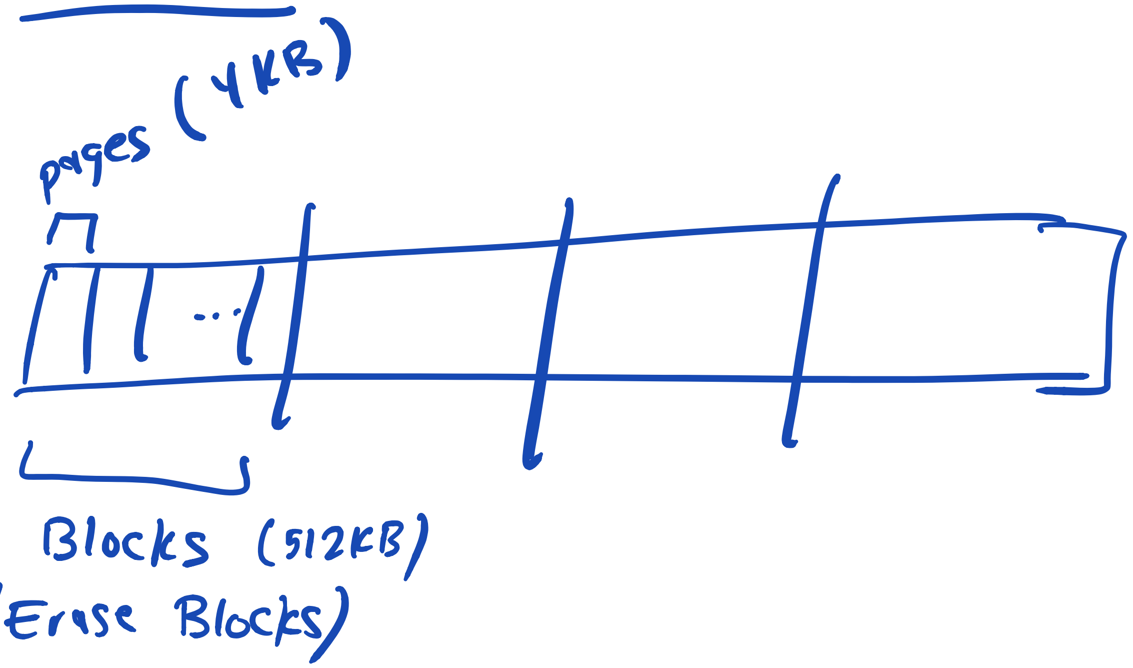


$$180\,\cancel{ms}\left(\frac{100\,\boxed{MB}}{\cancel{sec}}\right)\left(\frac{1\,\cancel{sec}}{1000\,\cancel{ms}}\right) =$$

$$\frac{180MB}{10} = \boxed{(18\,MB)}$$



SSDs:

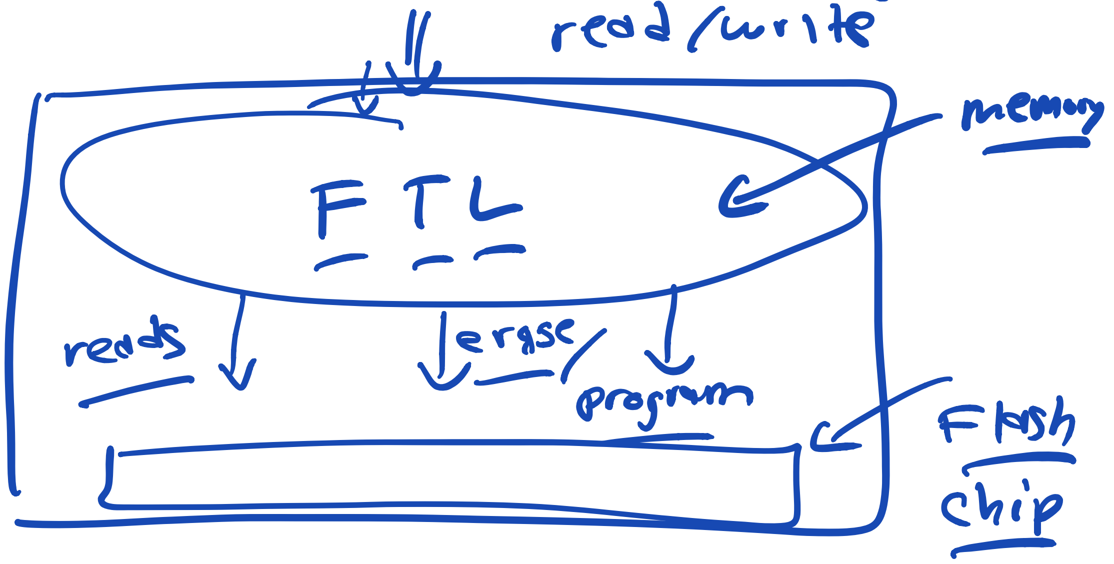


writing: ⇒ Erase Block (makes all pages valid)

⇒ Program each page once

worries: too many erases ⇒ wear out

same "block" interface read/write

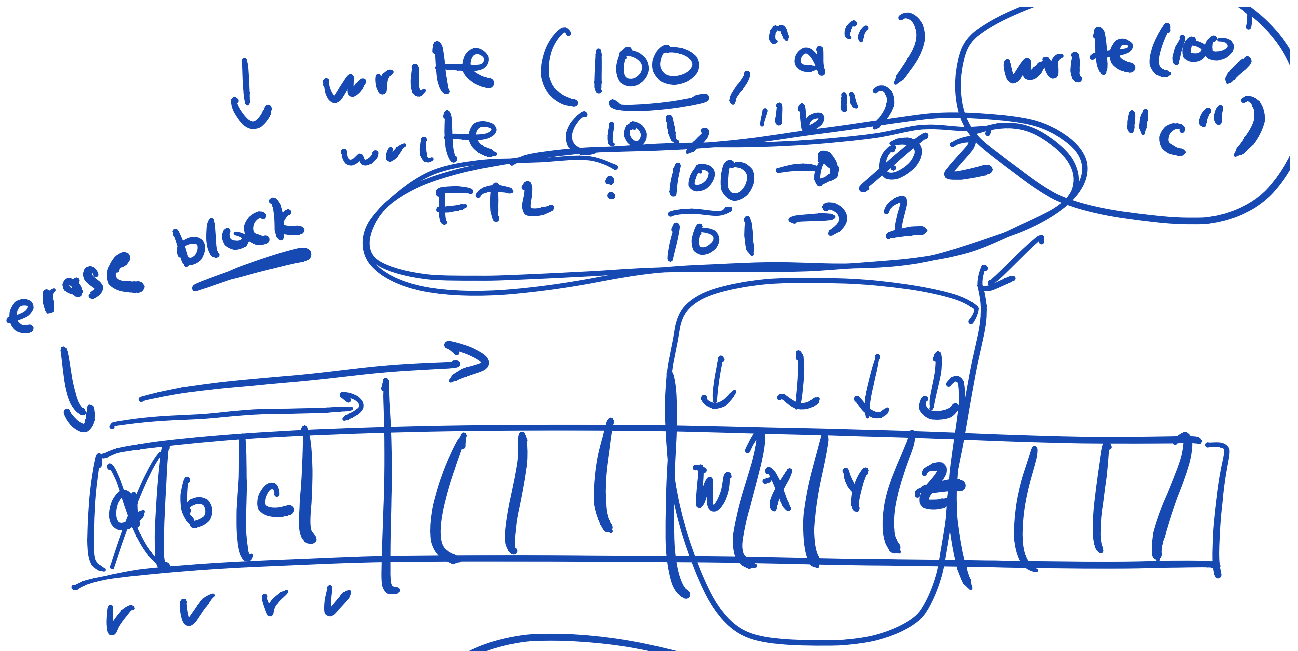


Flash Translation Layer:

⇒ Simplest "working" design

Logging of each logical block

+ mapping (logical blocks ⇒ physical pages)



concerns: logging

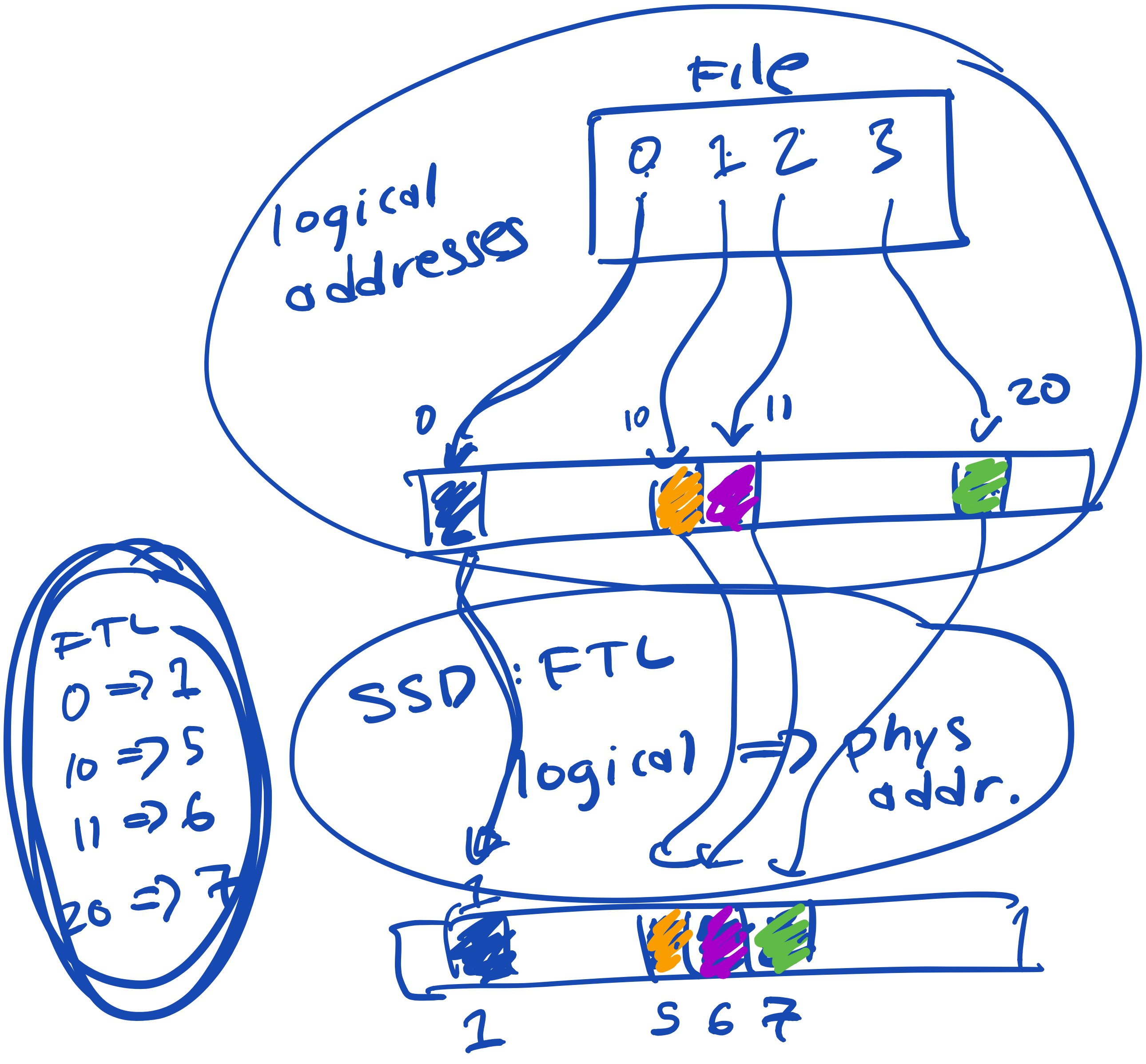
⇒ garbage collection
(background)

⇒ wear leveling:
move live blocks
elsewhere to
balance erase load
(wear)

Problem: FTL ⇒ too big
(one mapping / logical block)

too many mappings!

File : inode (+ indirect blocks) addresses (locations of data blocks)



Reduce: FTL size

=> FTL memory: cache of translations

=> only keep "most popular" translations in memory of device

=> cost : performance

e.g. random reads (to more blocks than there are translations that fit in memory)

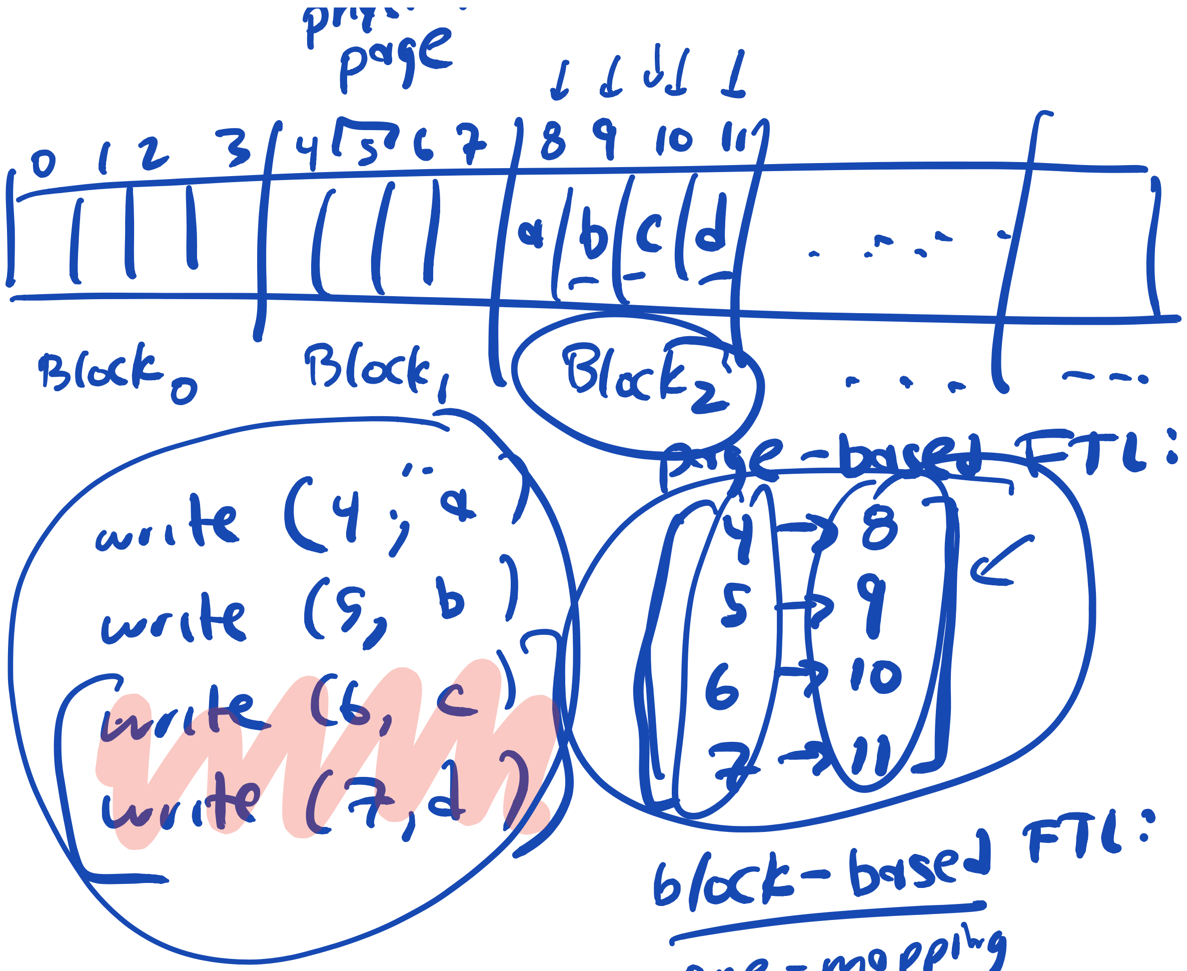
=> miss per access

=> 2x slower reads

=> More sophisticated mappings

=> Block-based

physical



block-based FTL:
one-mapping per block

phys block
4, 5, ..., ... → 2

Hybrid FTL : page mappings + block mappings

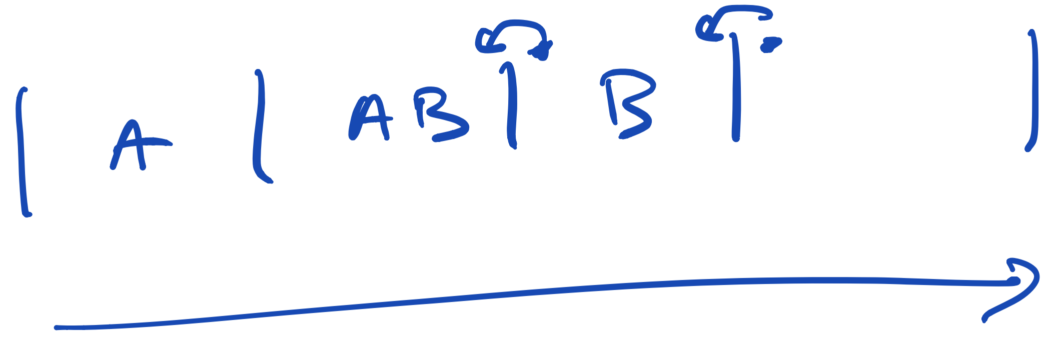
write (4, a)
write (5, b)

write (6, c),
write (7, d))

FTL : use page mappings at first

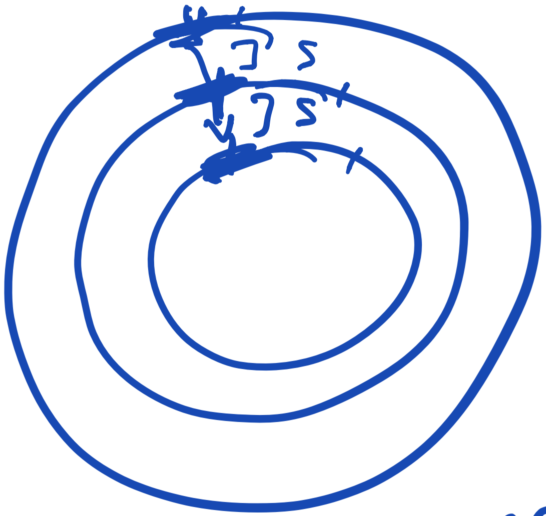
4 → 100
5 → 101
6 → 102
7 → 103

convert to smaller block-based mapping
(1 → 25)



⇒ Relation of S to R

⇒ Same request



SATF:

$\left(\underline{2S + 3R}\right]$

old:

$\underline{D \text{ time units}}$

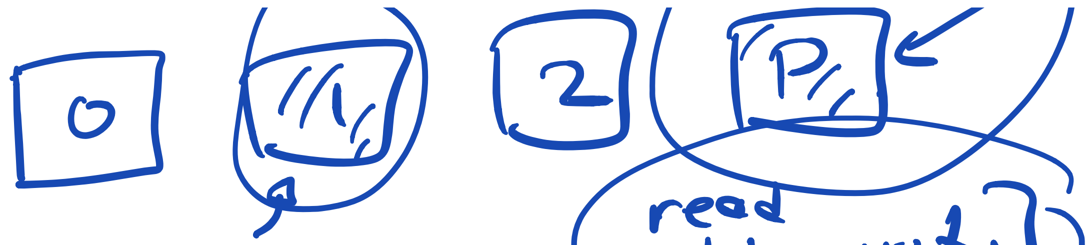
3D

new:

$\underline{\underline{\overset{Avg}{\text{Seek}}}} + \underline{\underline{\overset{Avg}{\text{Rotate}}}}$

$S + \frac{R}{2}$

RAID-4:



[Read old data, read old parity]

→ compute new parity

[write new data, write new parity]

model:

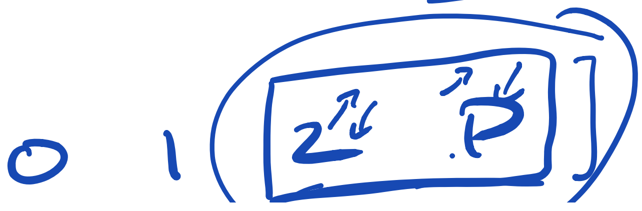
read old parity

write new parity

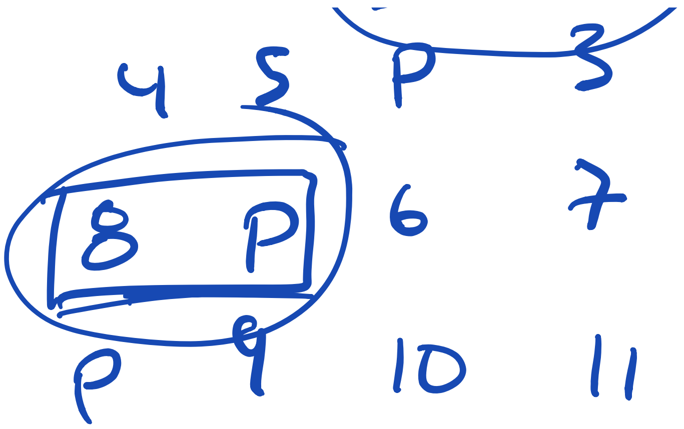
$$\overbrace{S + \frac{R}{2}}^{\text{READ}} \overbrace{+ \underline{\underline{R}}}^{\text{WRITE}} \Rightarrow S + \frac{3R}{2}$$

avg avg

RAID-5 : 12 writes



logical
2 writes



logical write:

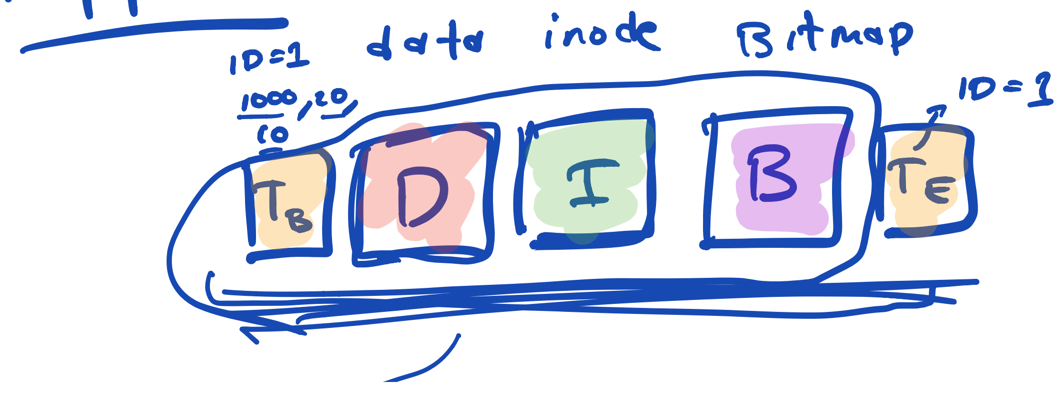
D [read   read] in parallel

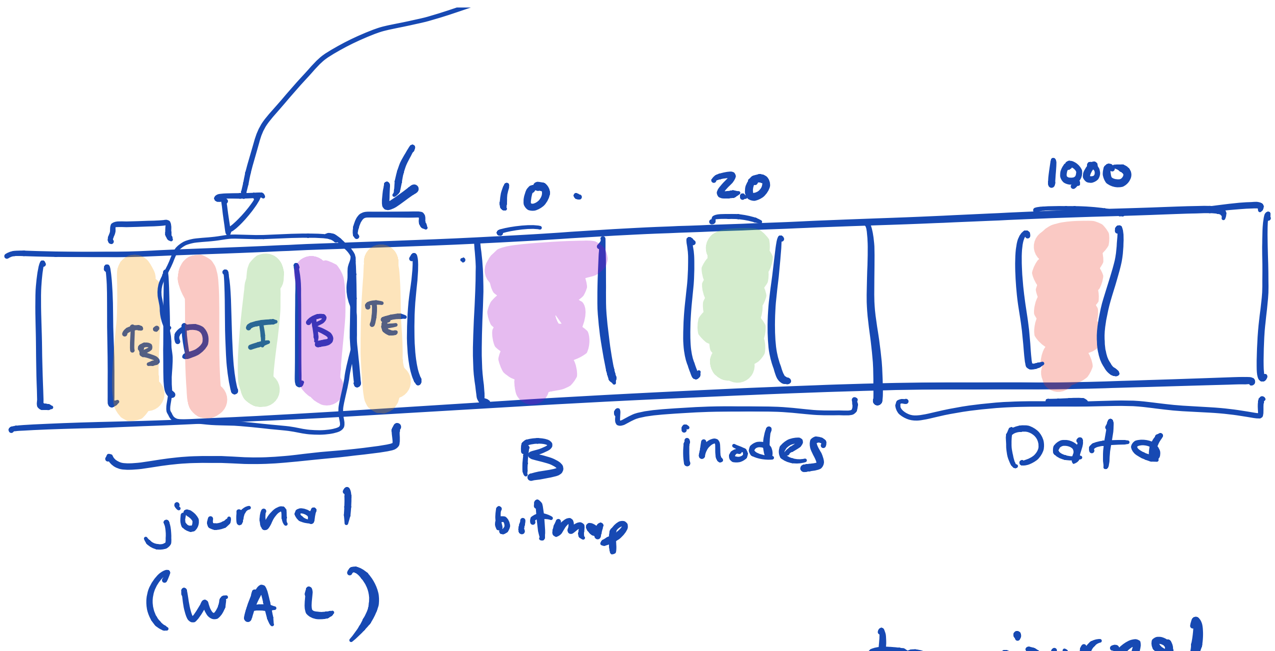
D [write   write] in parallel

=> logical write => 2 D × 6

=> 12 D

Append: write (data)





1) write everything to journal

$\Rightarrow$ T_B, contents all at once

wait

$\Rightarrow$ T_E : commits transaction

wait

2) update (overwrite B, D, I in place)