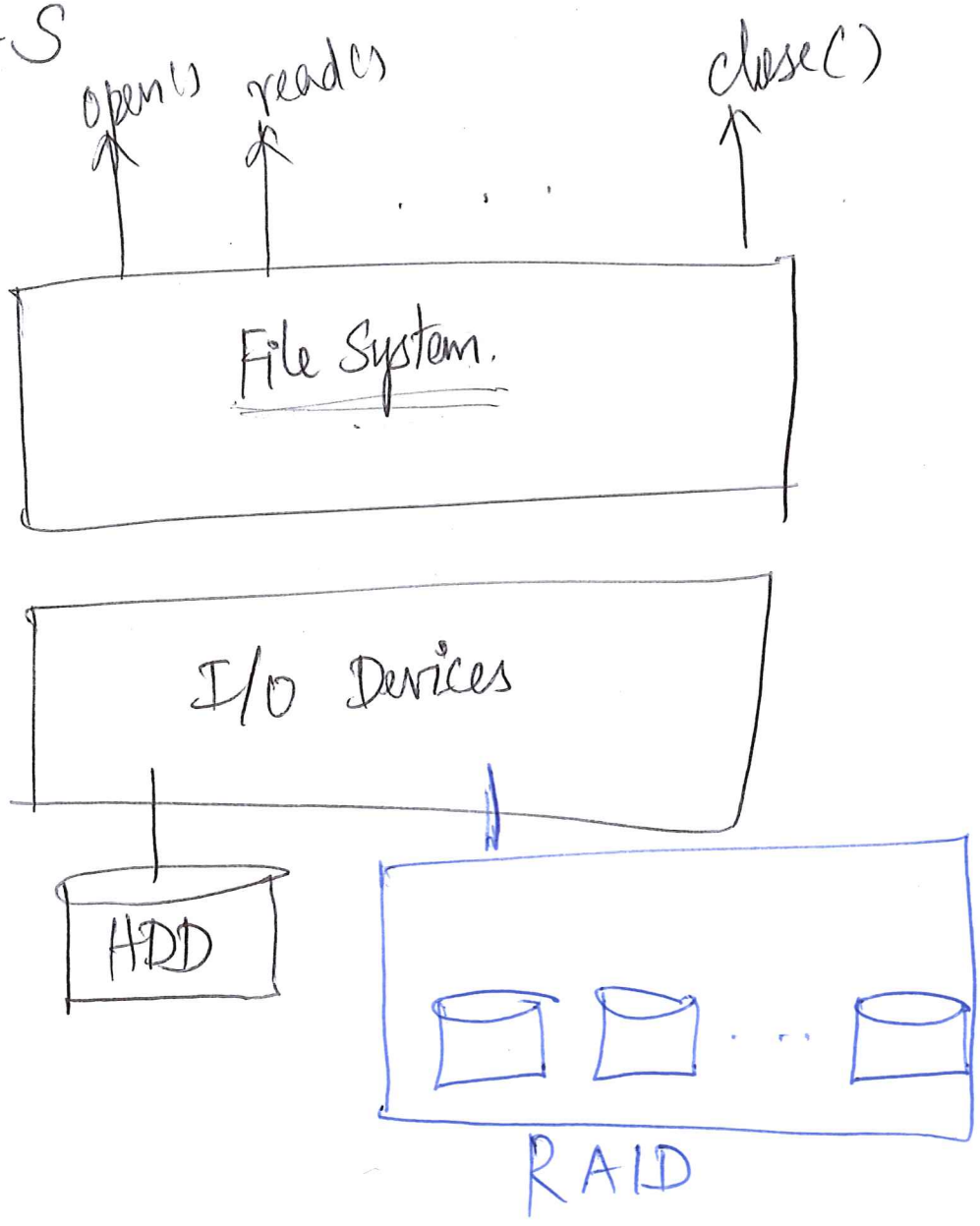
CS 537- Lecture 14

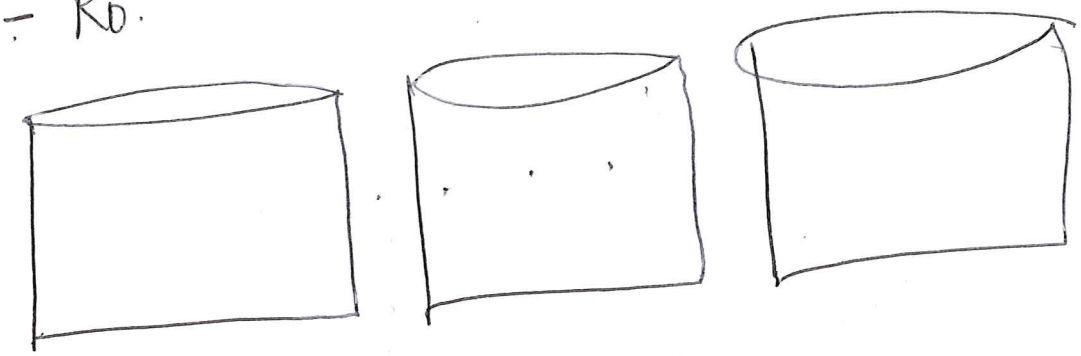
1. RAID
2. FFS
3. LFS

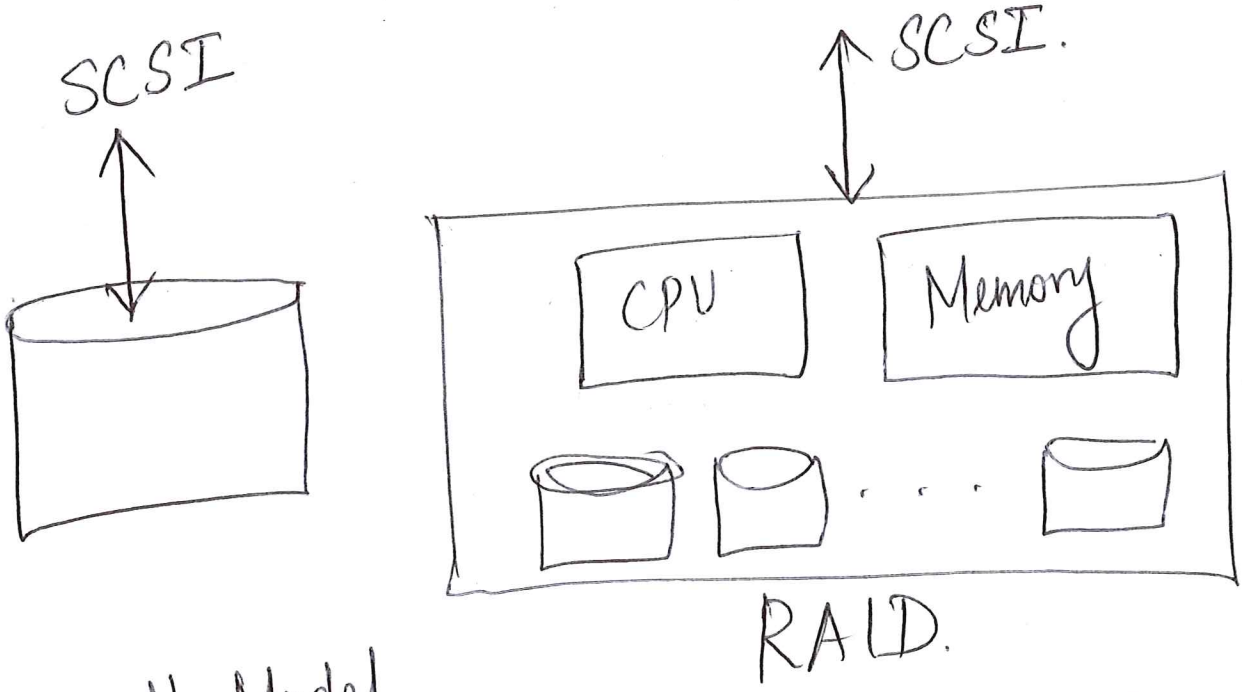


RAID

Redundant Array of Inexpensive Disks (Independent).

1) Reliability – R0, R4, R5
2) Performance:
3) Capacity – R0.



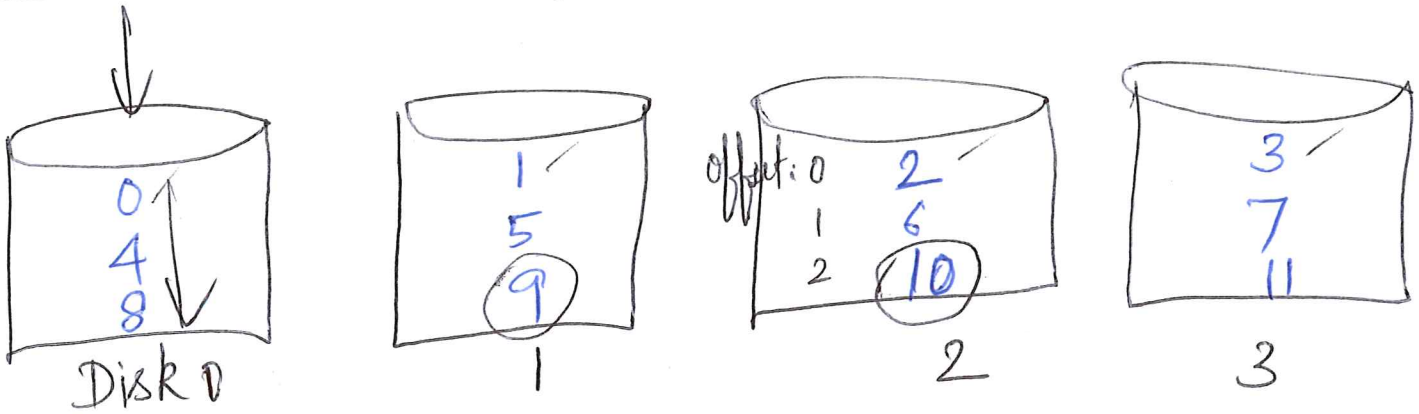


RAID.

Fault Model

FAIL - STOP < WORKING / FAILED

RAID-0 (striping)



disk # = Addr % NUMDISKS.

offset = Addr / NUMDISKS

chunk size

1 chunk = 4KB (1 block) ✓

1 chunk = 8KB (2 blocks)

D_0	D_1	D_2	D_3
0	2	4	6
1	3	5	7
8	10	12	14
9	11	13	15

	small chunk	big chunk
positioning time of disk head	more	less
intra-file parallelism	more	less

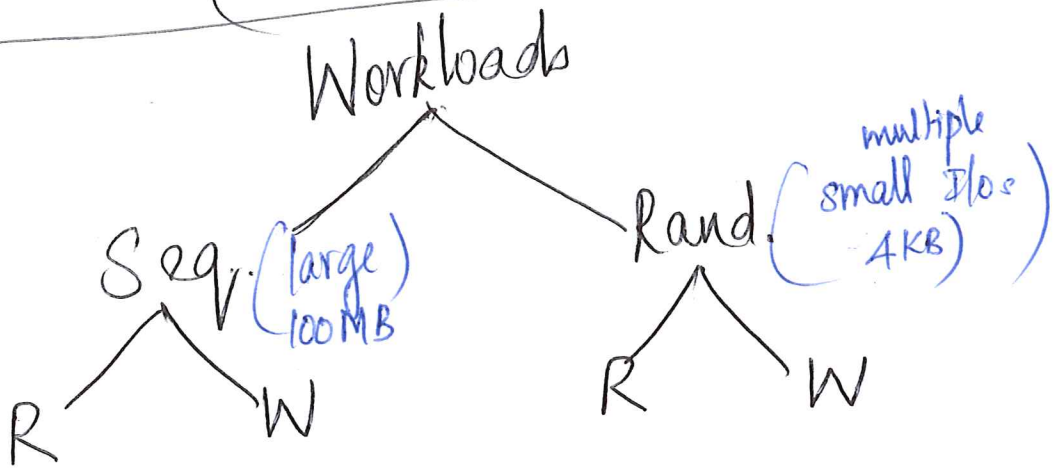
1) Capacity = N x B

disks # blocks/disk

2) Reliability = 0

3) Performance Analysis

1. Latency of a single request
2. Throughput of many requests. (steady-state throughput)



Data transfer Rate

Seq — S MB/s

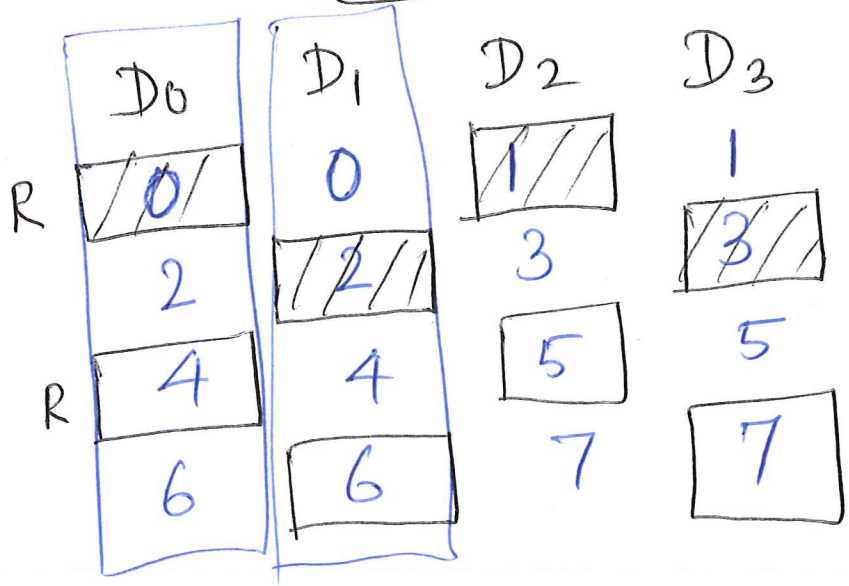
Rand — R MB/s

T = Time to read/write a block on a single disk.

RAID-0 Analysis

1. Latency
   - R = T
   - W = T
2. Throughput
   - Seq
     - R = N · S MB/s
     - W = N · S MB/s
   - Rand
     - R = N · R MB/s
     - W = N · R MB/s

RAID-1 (Mirroring)



capacity = $\frac{N}{2} \times B$.

Reliability = $\frac{N}{2}$ (if lucky)

= 1

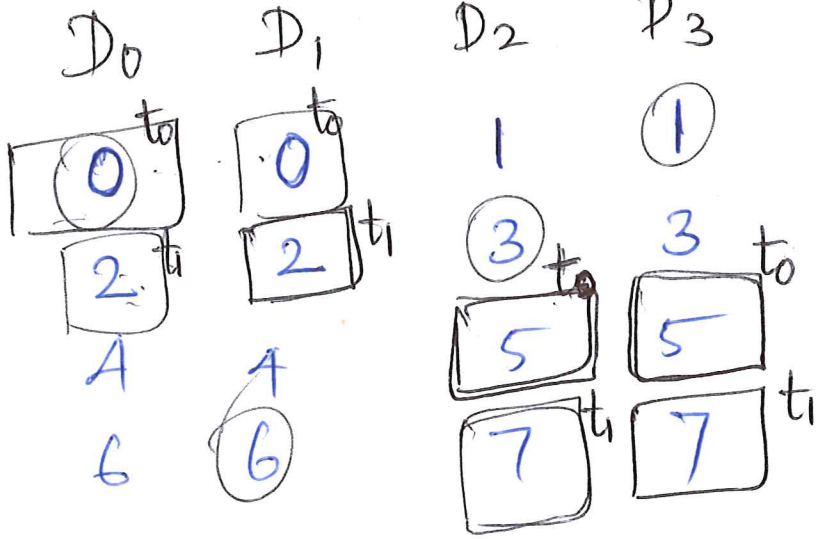
RAID-1 Analysis

1. Latency $\begin{cases} R = T \\ W = T \end{cases}$

2. Throughput

1. Sequential
   - Reads = $\frac{N \times S}{2}$ MB/s.
   - Writes = $\frac{N \times S}{2}$ MB/s

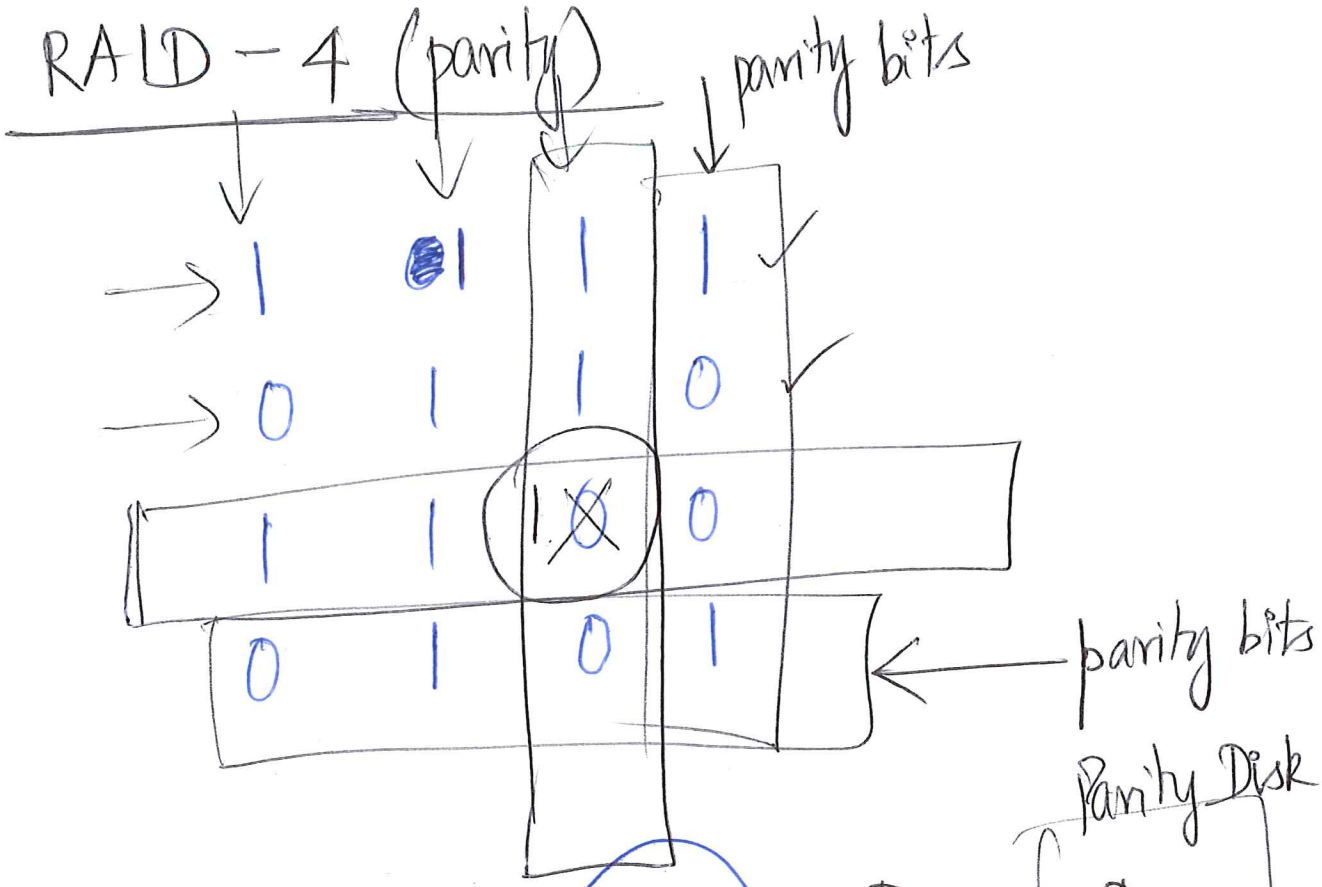
2. Random
   - Reads = $N \times R$ MB/s
   - Writes = $\frac{N \times R}{2}$ MB/s



Write
0-3.

Random write : 0, 2, 5, 7

RAID – 4 (parity)



RAID-4 Analysis

capacity = $(N-1) \times B$

reliability = 1

Latency < Read = T
Write = 2T

How writes happen?

D_0	D_1	D_2	D_3	D_4
0	1	2 (Cold)	3	P (Pold)

$0 \oplus 1 \oplus 2' \oplus 3 = P'$ (Cnew, Pnew)

How to compute parity?

Write to block 2

1. Read Blocks 0, 1, 3.
2. ~~Write~~ Compute new parity.
3. Write blocks 2' and P'

} Additive parity

Subtractive Parity

$$P_{new} = (C_{old} \oplus C_{new}) \oplus P_{old}$$

$$(0 \oplus 1) \quad 1$$

$$\Downarrow$$

$$1 \quad \oplus \quad 1 \Rightarrow 0 \quad (P_{new})$$

$$=$$

1. Read Cold
2. Read Pold

} Parallel

3. (Pnew) = (Cold ⊕ Cnew) ⊕ Pold // compute new parity.
4. Write Cnew
5. Write Pnew.

} Parallel.

Throughput

1. Seq
   - Read = $(N-1)\,S$ MB/s
   - Write = $(N-1)\,S$ MB/s

$C_0' \oplus C_1' \oplus C_2' \oplus C_3' = P_{0-3}'$

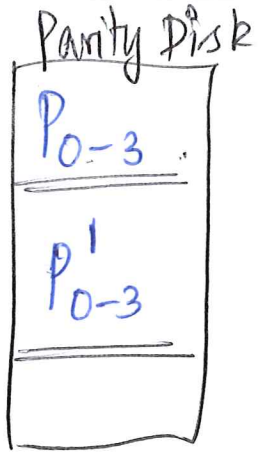
↓

new content of C_0.

2. Rand
   - Reads = $(N-1).\,R$ MB/s
   - Writes = $\frac{R}{2}$ MB/s.

t_0: Read 0, 5, 10, 15,

t_1: Write 0',



RAID-5 (Rotated Parity)

D_0	D_1	D_2	D_3	D_4
0	1	2	3	P_{0-3}
4	5	6	P_{4-7}	7
8	9	P_{8-11}	10	11
12	P_{12-15}	13	14	15
P_{16-19}	16	17	18	19

capacity = $(N-1)B$

reliability = 1

latency < R = T, W = 2T

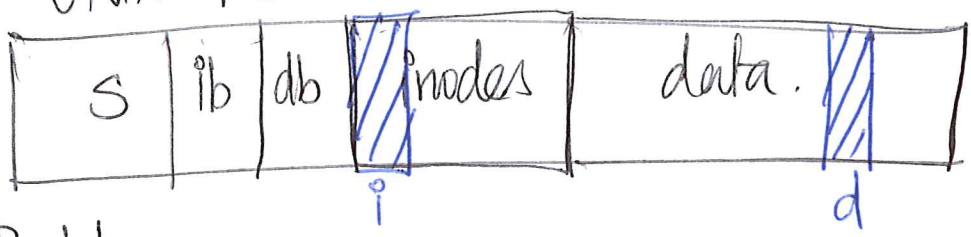
throughput < Seq < R = $(N-1)\cdot S$, W = $(N-1)\cdot S$

throughput < Rand < R = $N \cdot R$, W = $\frac{N \cdot R}{4}$ MB/s.

FFS

Fast File System.

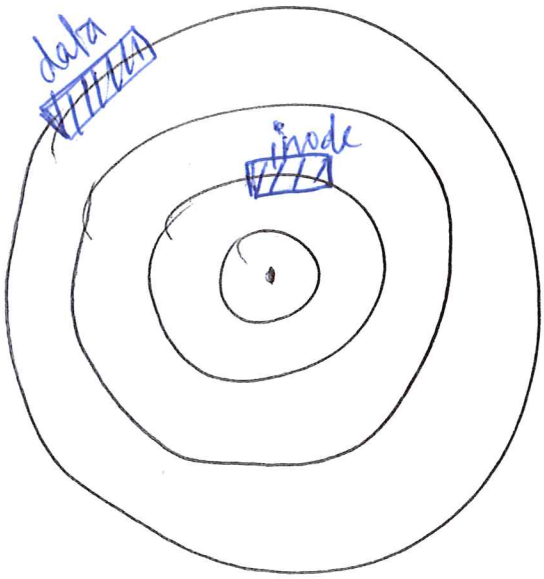
UNIX FS



Problems

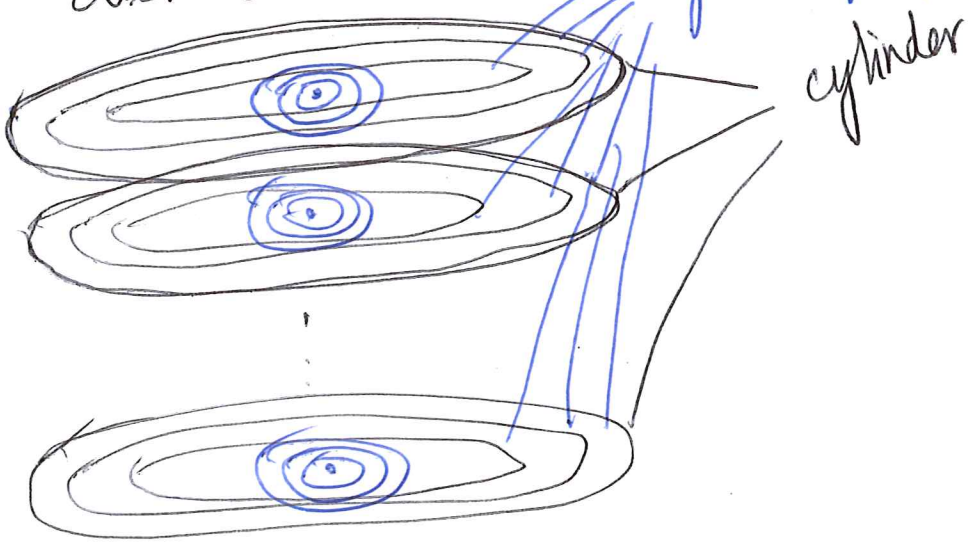
read a file

1) read inode
2) read data blocks

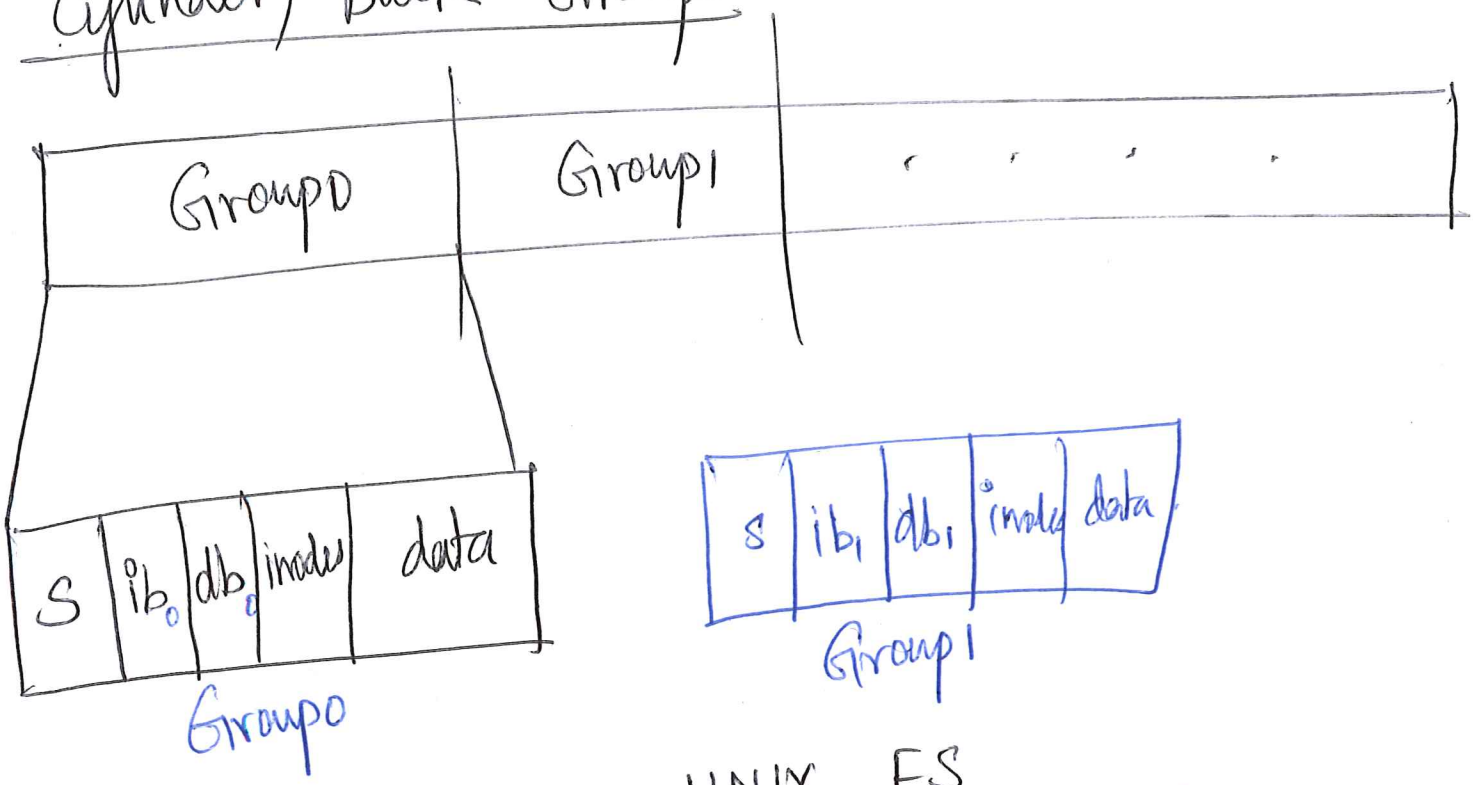


1. Performance
2. UNIX FS - Small blocks (512 bytes)
   → reduce internal frag.
   → increases positioning costs.

Soln: FS disk-aware!



Cylinder / Block Groups



APIs – same as UNIX FS
open(), read(), write(), close().

Allocation Policies

Rule: Keep related stuff together.
" unrelated " apart.

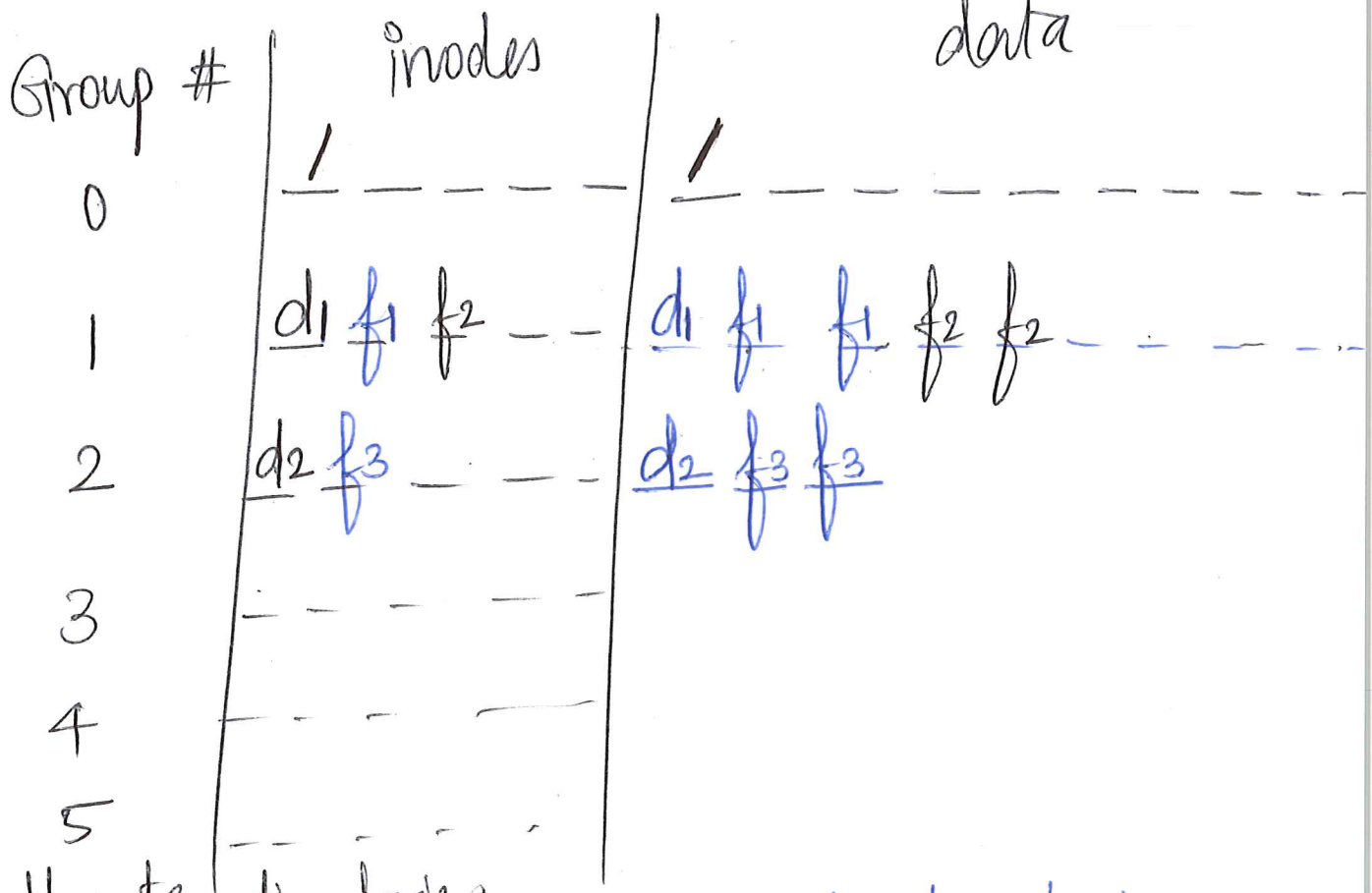
What is related?

1) a file's inode + data ⟹ same group.
2) files within the same directory.

What is unrelated?

diff. directories.

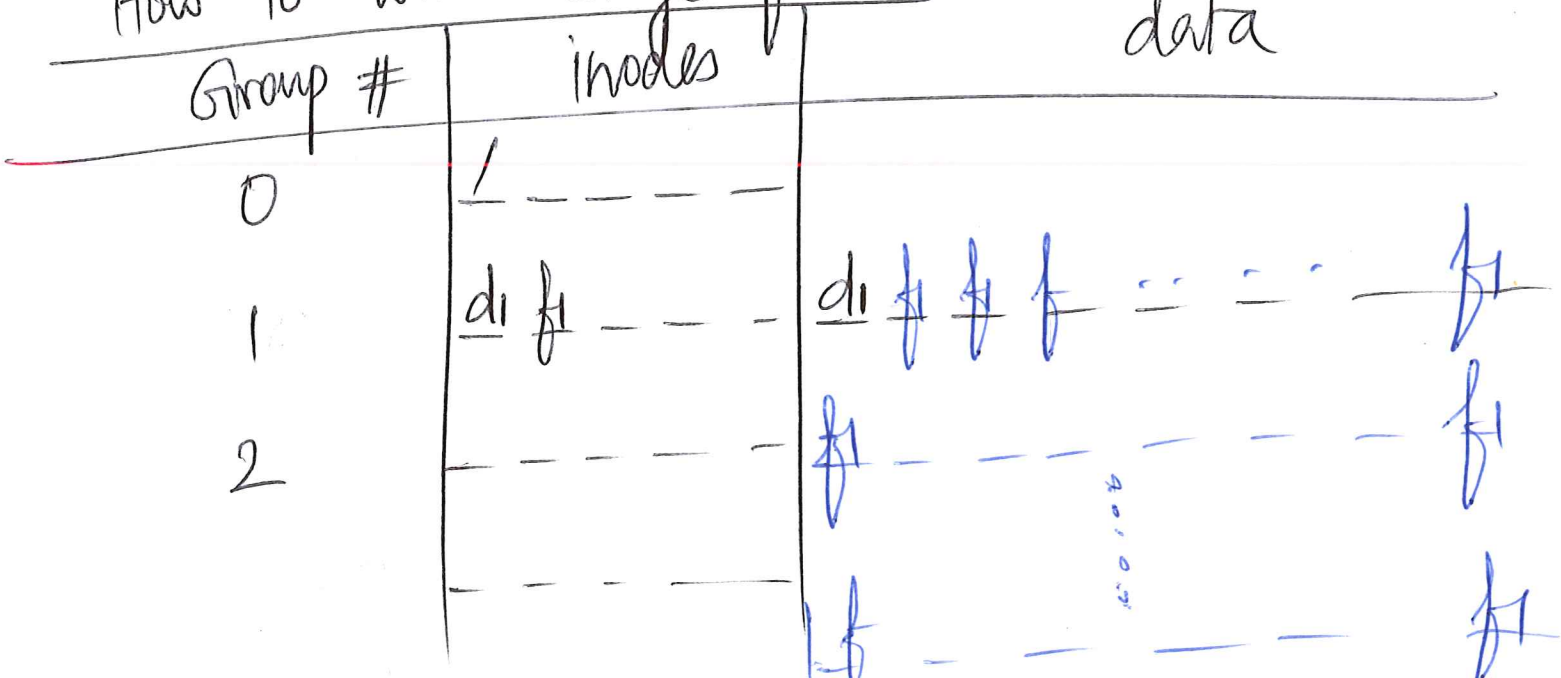
/d1/f1 /d1/f2 /d2/df3



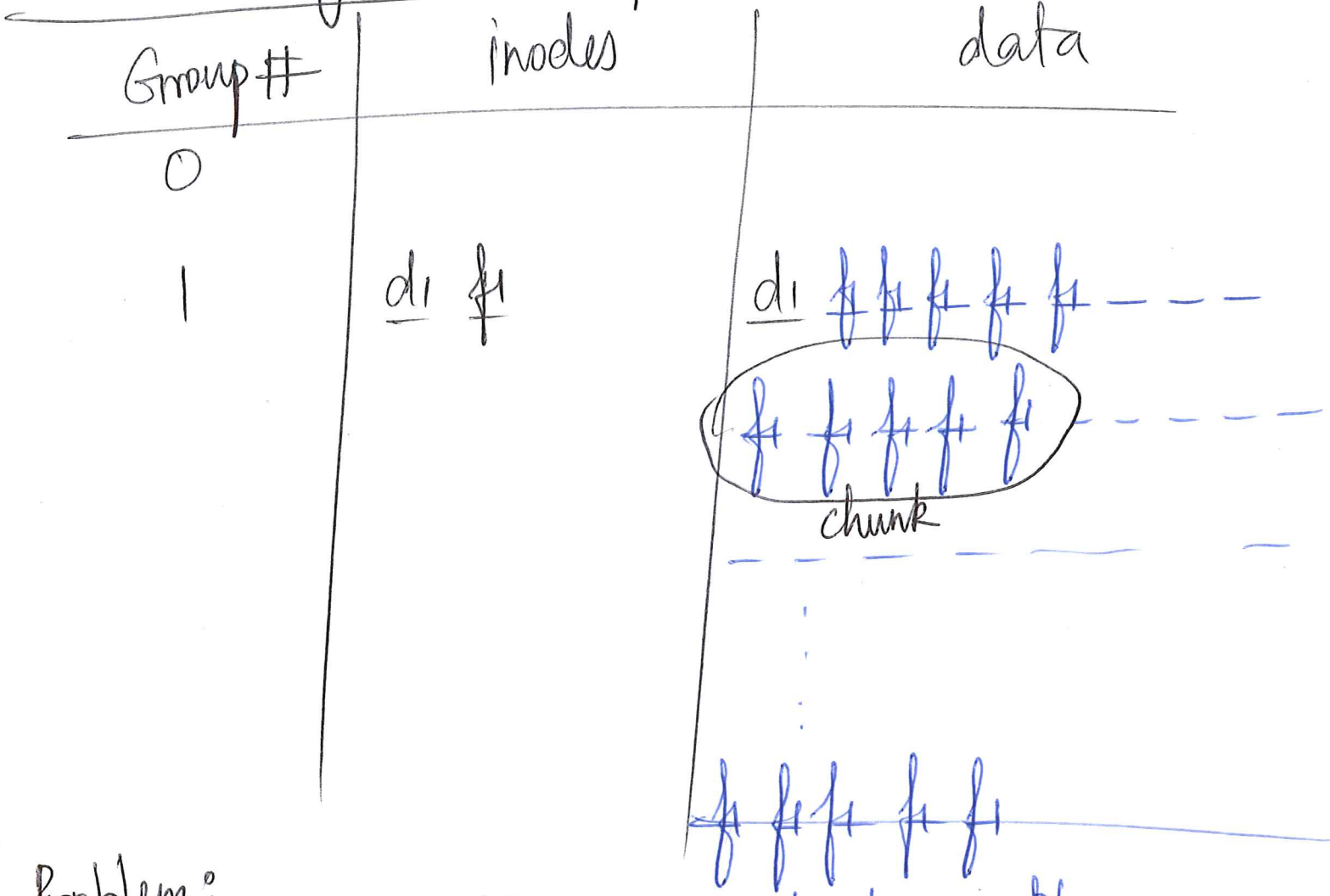
To allocate directories

1) which groups have low # of directories
2) " " " a high # of free inodes

How to write large files?



FFS- Big File Exception



Problem:

Perf. of seq. file access of large files

Positioning Time + Data transfer time

(Seek + Rotation)

10 ms

10 ms

Data transfer rate = 40 MB/s

1 s → 40 MB

10 ms → $\frac{40\,MB \times 10\,ms}{1\,s} = 40 \times 10^{6} \times 10^{-3}$

$= 400 \times 10^{3} = 400\,KB$

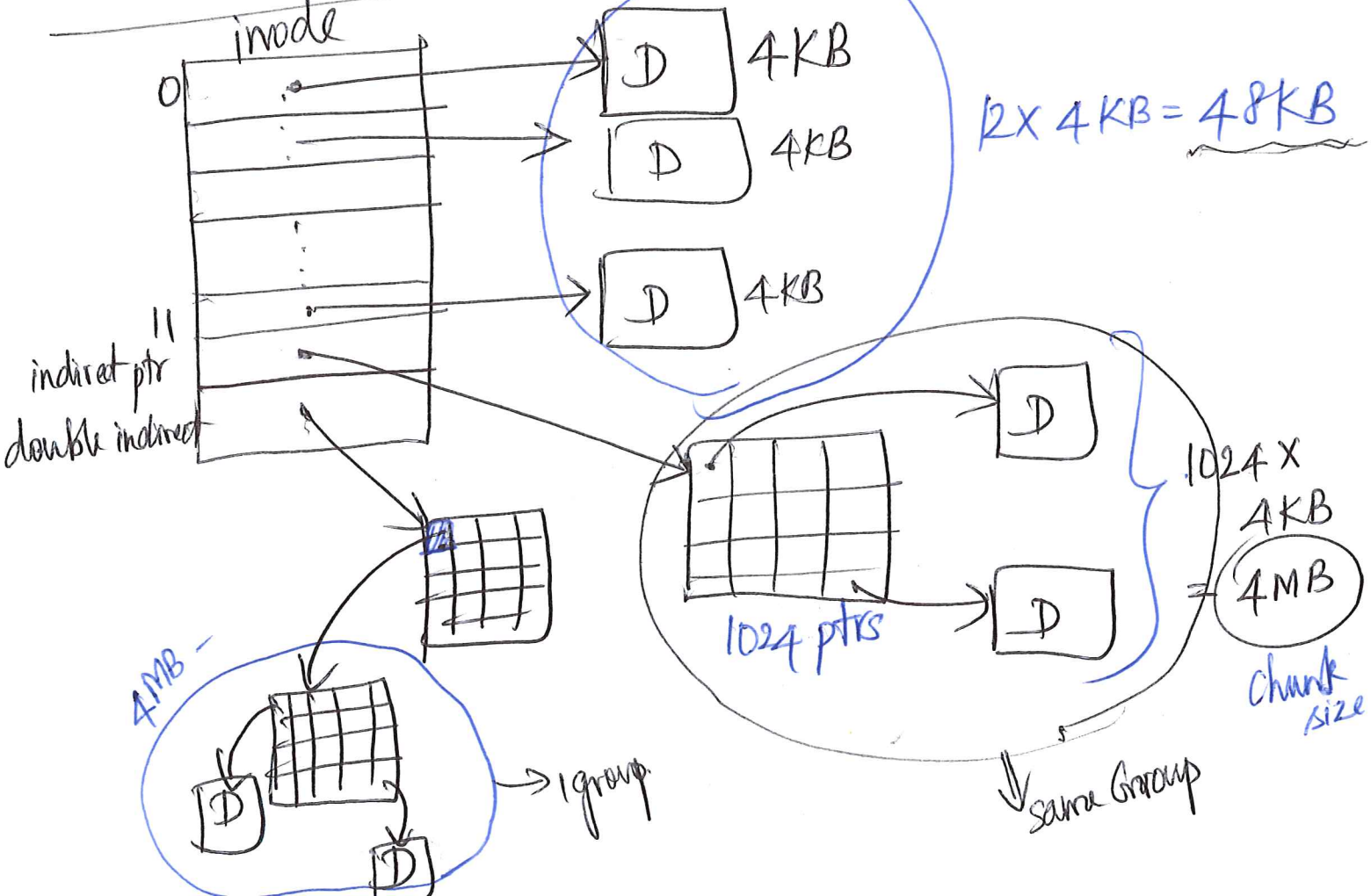
Positioning	Data transfer
10% time	90% time
10ms	90 ms

$$1s \longrightarrow 40\,MB$$

$$90ms \longrightarrow \frac{40\,MB \times 90\,ms}{1s}$$

$$= 3600 \times 10^{3} \simeq \boxed{3.6\,MB}$$ (chunk size).

FFS- did NOT use amortization.



LFS
Log-structured File System.

Observations

1. Systems have larger memories
   - Reads are serviced from memory.
   - Disk traffic - writes *

2. Large gap. Seq I/O Rand I/O

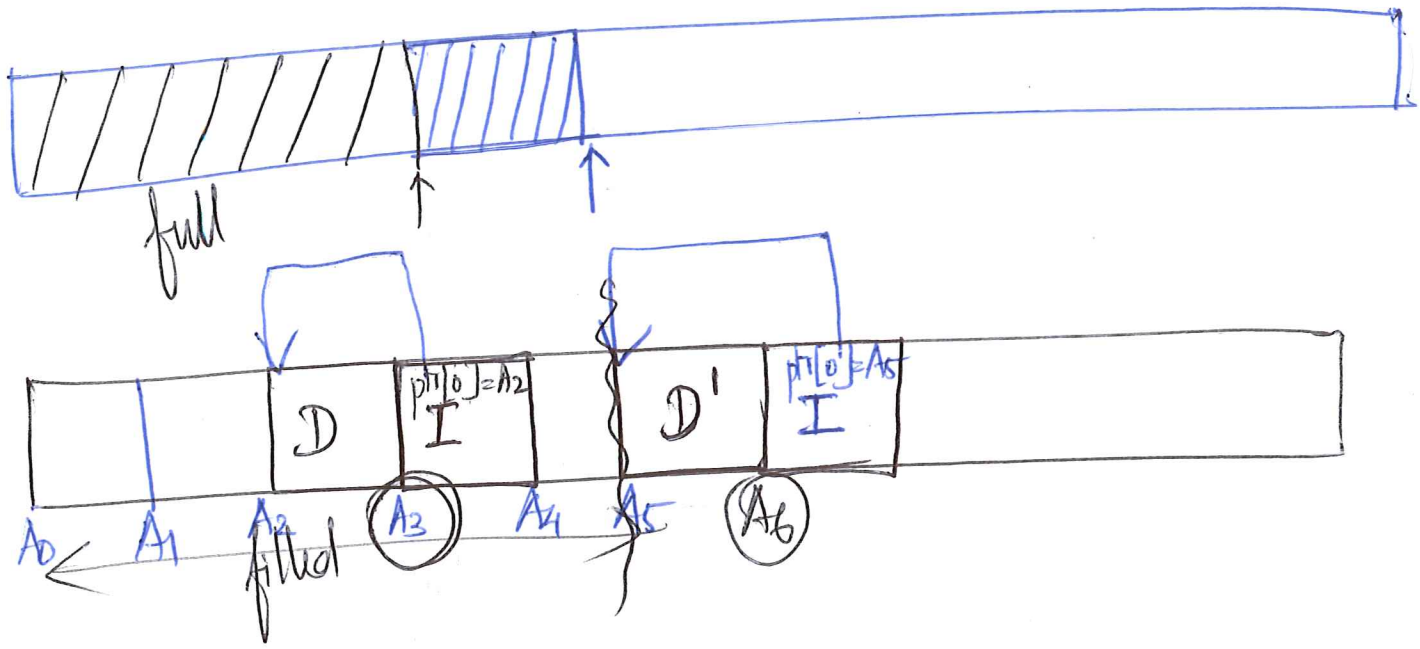
   S >> R

   Random writes → Seq writes

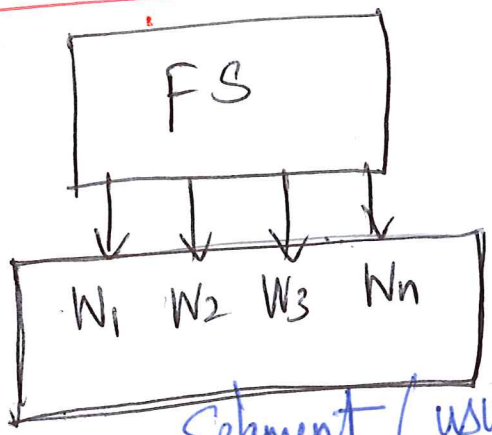
3. Existing FS perform badly on common workloads
   e.g. file creation.
   1. inodes < dir, file
   2. bitmap < inode, data.
   3. data < dir, file.

4. FS → not RAID aware.
   "small writes problems"

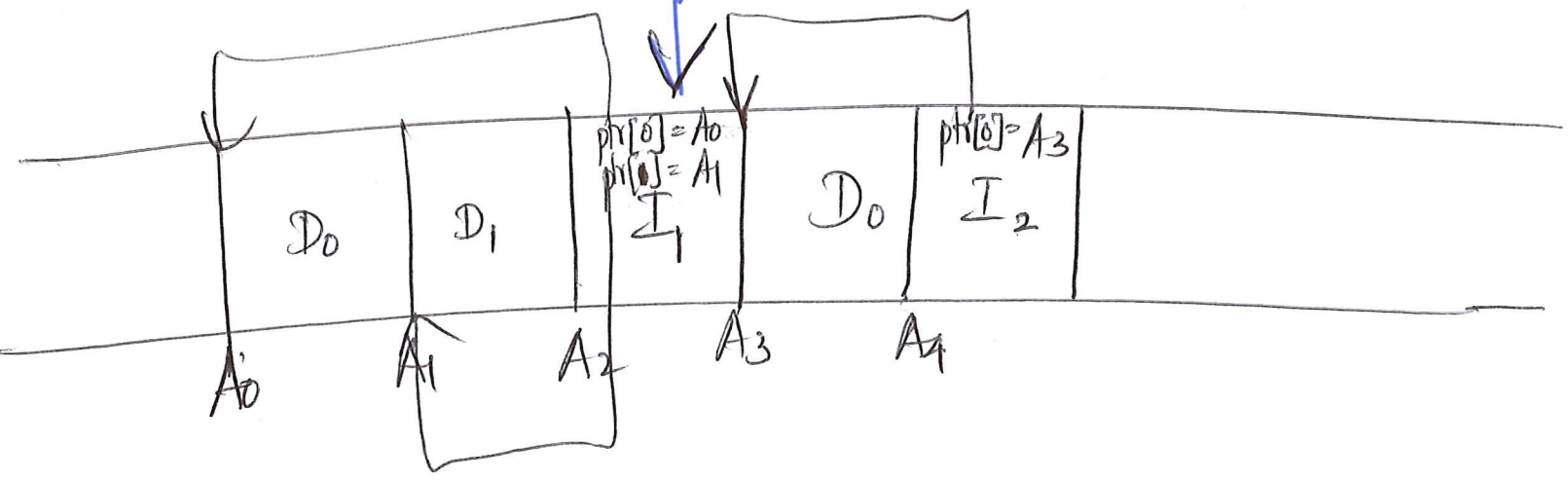
IDEA:

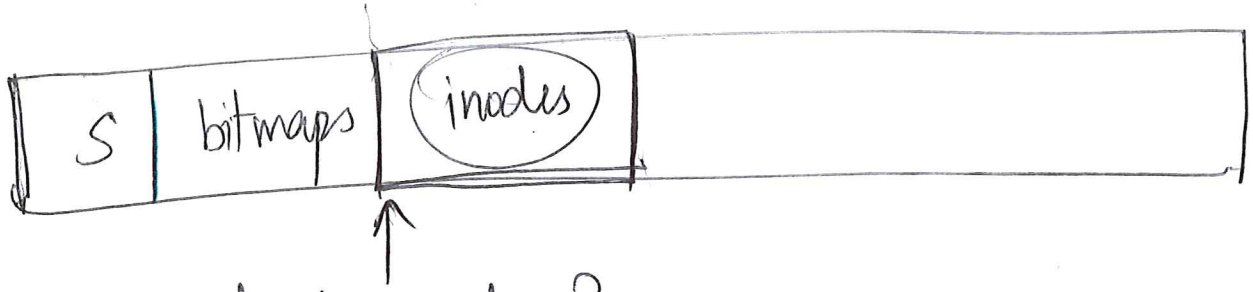


WRITE BUFFERING.



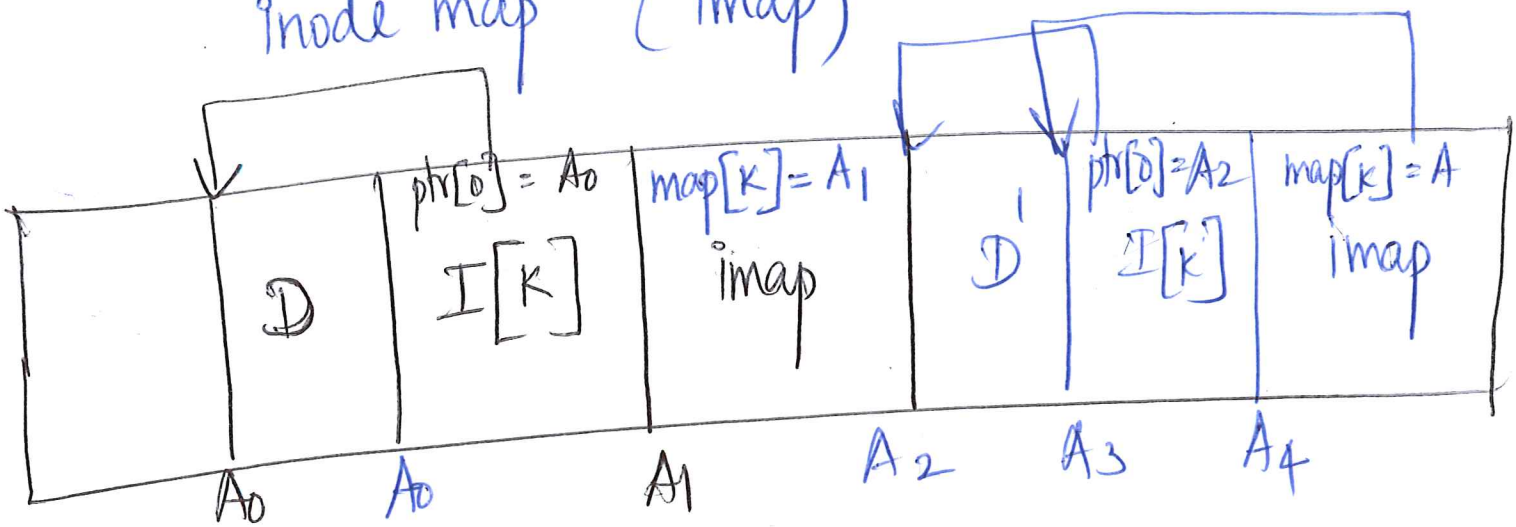
Segment (usually in MB)



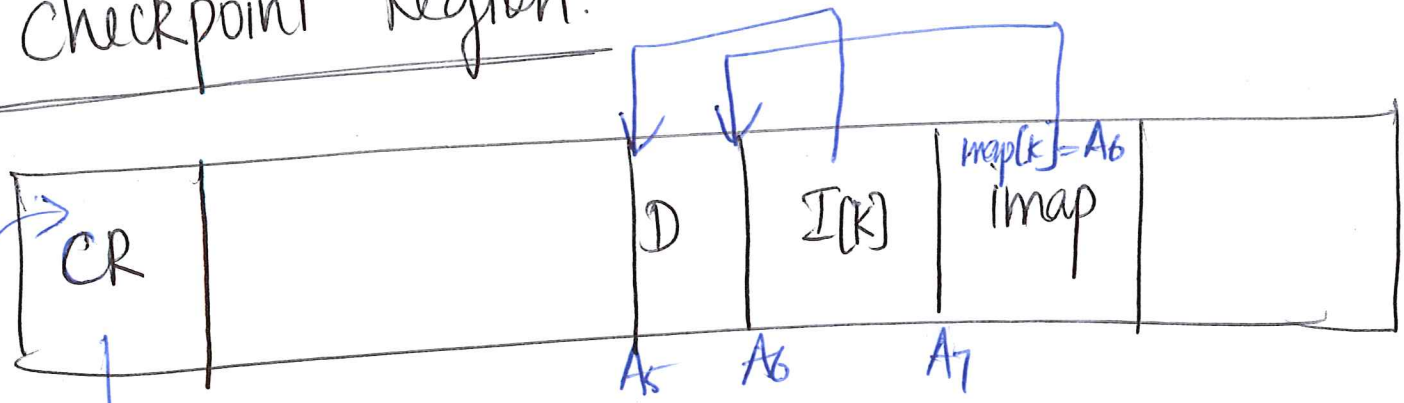


How to find inodes?

inode map (imap)



Checkpoint Region.



fixed location.

imap[K .. K+n] = A7

Append