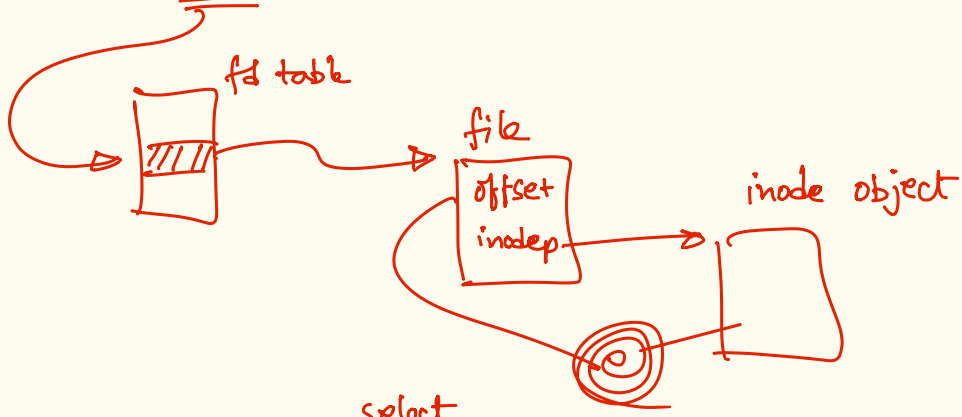
Lecture 23 — file system consistency / reliability

write (fd, buffer, size)



- allocate/~~reserve~~ (select) datablock
- update inode w/ datablock info./size/timestamp
- write content to selected datablock
- write/set the (data) block bitmap

the crash consistency problem.

- if all 3 writes happen — all good!
- if NOT, ——— ??

FS	inode update	block bitmap	write datablock	
consistent (C)	✓	✓	✓	
consistent (C)	×	×	×	
unused write (C)	×	×	✓	
inode w/ garbage/leaked data (FC)	✓	×	×	—
datablock leak (IC)	×	✓	×	
garbage in datablock (FC)	✓	✓	×	
datablock can be overwritten (FC)	✓	×	✓	—
datablock leak (IC)	×	✓	✓	

(*) FS promise is of consistency !

↳ 3/multiple actions on disk (metadata ~ FS, data ~ disk) have to be atomic

all or nothing

(i) fsck — file system consisteny check

- reactive
- to some extent detect inconsistency
- given a disk w/ the File system used to manage it

check for consisteny

- reads metadata — inode
  bitmaps
  & disk state for checks

- superblock ~ FS features — inode size
  start inodes / bitmaps

#inodes = inode region / inode size

- datablocks ~ all datablocks are accounted for!
  - some inode points to a datablock & block bitmap is set.

- inodes — size ~ set of datablocks
  ~ inode bitmap set — flag in inode / used / unused
  - links : multiple files that map to same inode
    refcnt.

~ bad blocks w&r consistent
- directory structure
  ∴ } are these 2 entries present

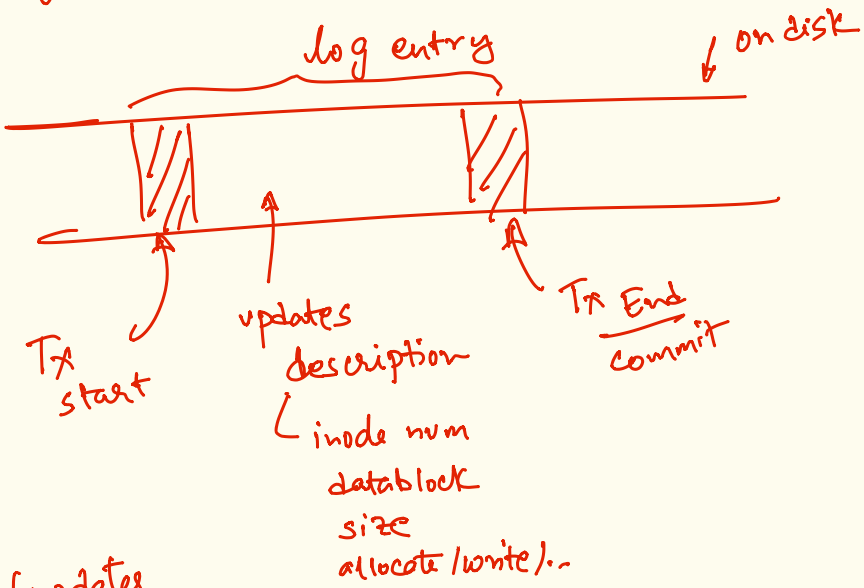
(ii)

~ all-or-nothing semantics on all updates of a FS functionality

- transaction semantics

write-ahead logging / journalling

- each update/write is maintained in a separate log into
- log is replayed for actual FS update.



on write/updates of FS

- start a Tx (if needed)
- update the log
- add Tx commit (if needed)
- flush log
  - periodic
  - on commit
  - lazy ~ on operation related to files in the log.

(3) reliability / performance

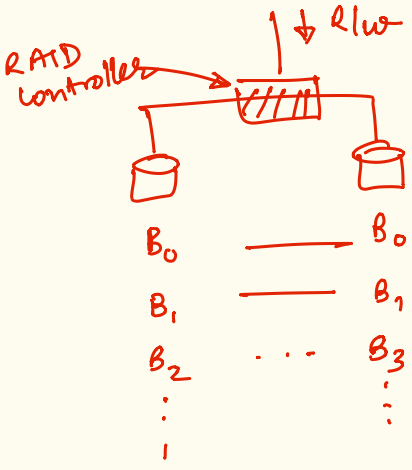
RAID ~ redundant array of independent disks.
inexpensive

hw or/and sw based reliability mechanism.

(i) redundancy / duplicates

(ii) striping / partitioning (performance)

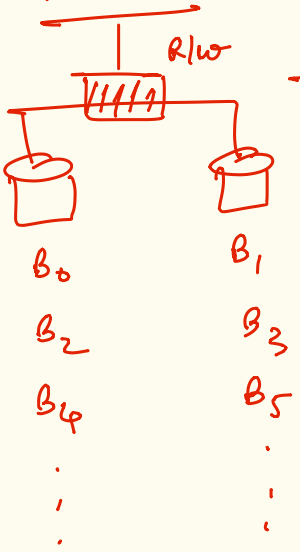
RAID 1



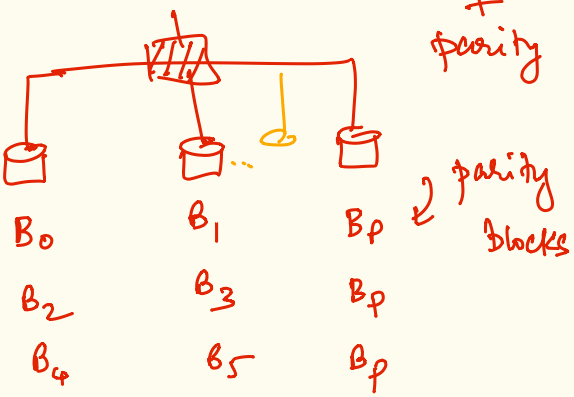
- costly!
- double resources
- writes are slower.

RAID 0 (performance)

striping



RAID 4 striping + parity



RAID 5 (balancing load on parity disk)