CS219 Lecture 35

Labquiz 4 today! 1.45pm.

file system consistency

~ reads are great for caching

writes cause a consistency problem.

- multiple copies
- single FS write syscall ⇒ multiple writes to disk.

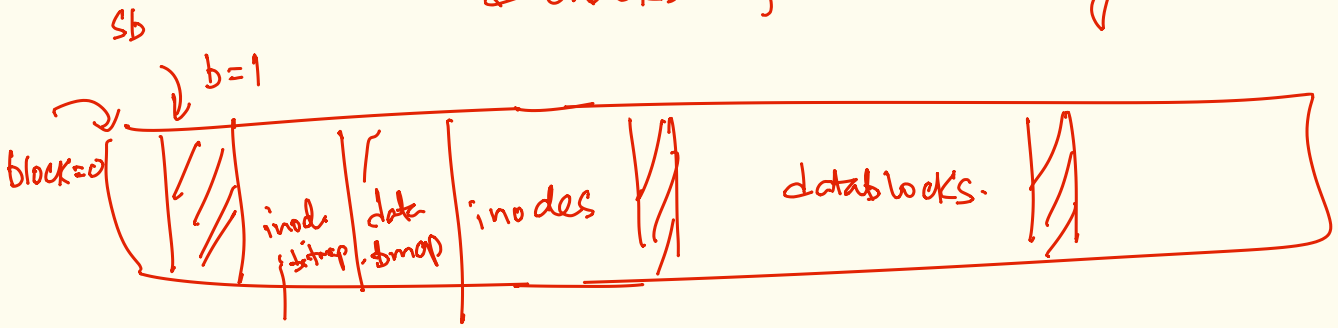
		inode update	data bitmap update	data block update
C		✓	✓	✓
C		✗	✗	✗
data loss!	IC	✗	✗	✓
inode state ≠ bitmap state	IC	✓	✗	✗
leaky block	IC	✗	✓	✗
garbage data	IC?	✓	✓	✗
	IC	✓	*	✓
leaky block	IC	*	✓	✓

(*) File consistency : ~~FS~~ file system state on disk has metadata matching w/ data & state of device.

FS promise — file abstraction
persistence, consistency, reliability, efficiency.

① fsck — file system consistency check.

- reactive, after-the-fact check!
- Q. is the FS consistent.
- looks/scans a disk device which is managed by a FS.
  reads the disk raw (block by block)
  & interprets FS state on it
  & checks for consistency.



superblock : FS size ≤ device size
disk

check integrity of superblocks

#inodes $\stackrel{?}{=}$ inode region/size of inode

inode bitmap size $\stackrel{?}{=}$ #inodes

~~disk~~ db bitmap $\stackrel{?}{=}$ #datablocks

- datablocks : are all datablocks accounted for:

(i) is a dB in inode but not set in bitmap?

(ii) is a dB set in bitmap but not in any inode?

- inodes: inode bitmap $\stackrel{?}{=}$ inode used.

- bad blocks — ~~R/W~~ $W + R = W$

- directories — loops

if : } first 2 entries

- inode datablocks out of range..

- size $\stackrel{?}{=}$ # data blocks.

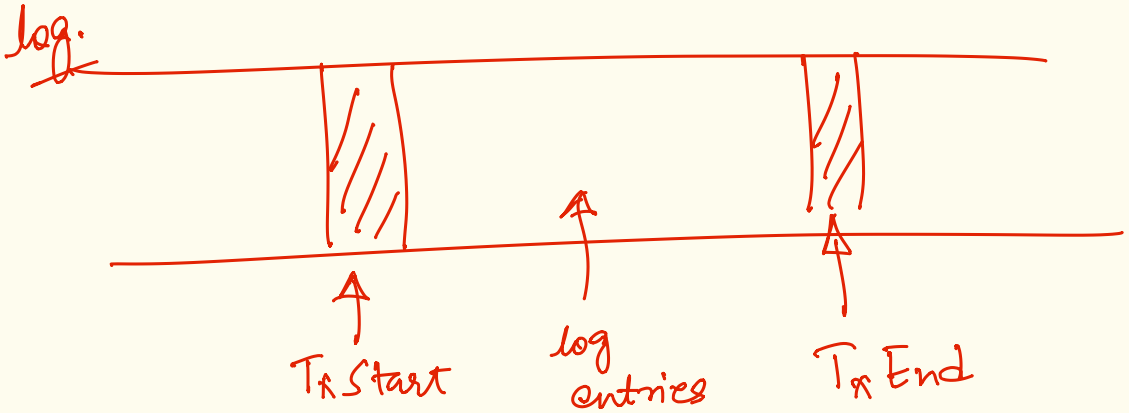
---

fsck (-ve)

- reactive
- cannot ~~can't~~ catch all inconsistencies.

(2) journaling / write-ahead logging

- info. about writes/updates (metadata, data) stored in a log.
- log has transaction semantics.
  - all or nothing



FS on write

~ Tx Start (if needed)

~ update/write log entries to journal
  - update inode 23 update into
  - update db bitmap 73 update into.
  - update datablock 2023 update into (data)

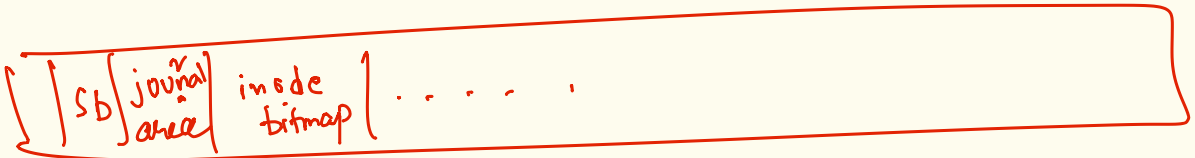
~ commit (Tx End)

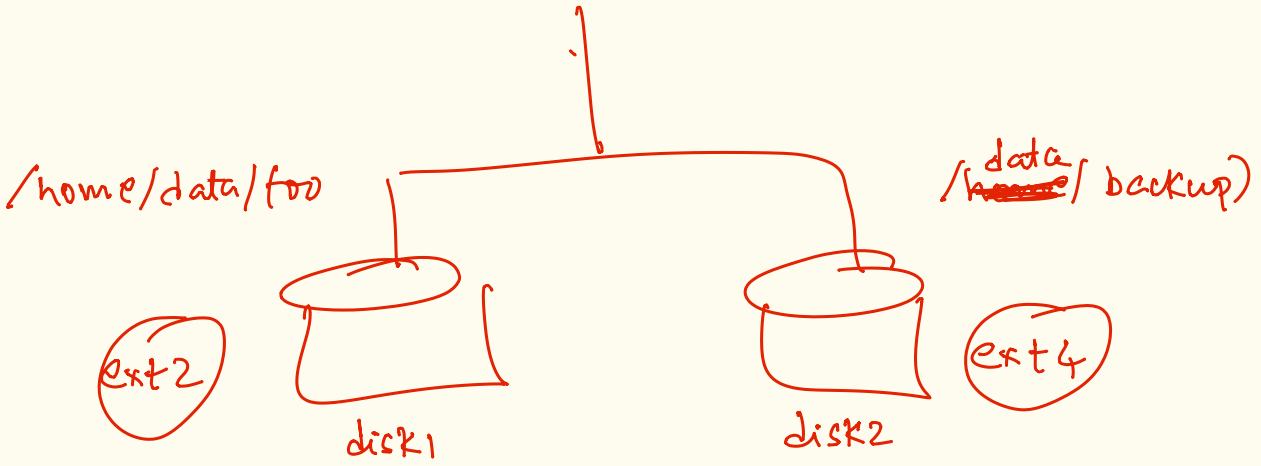
~ periodically or on read flush the log to update FS metadata.

~ post flush clear the log entry

Q where to store log!

- to store on disk





cp /home/data/foo /data/backup