11/29/23

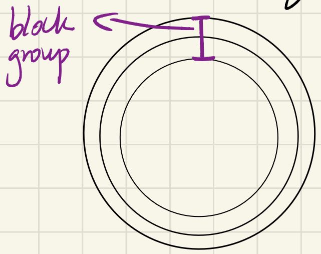
Log Structured File System (LFS)

→ optimize for write performance on disk, utilize peak disk bandwidth.

→ more memory ⇒ larger buffer cache ⇒ serves more read w/ cache

→ disk traffic will mostly be writes

→ FFS. (block group placement)



shorter seek time when access sectors within the same block group (less movement)

each block group has its own inode table, inode bitmap, data bitmap, data blocks

☆ shorter seeks but still slow

→ LFS : amortize seek time even more by doing large sequential writes

How can we almost only do large sequential writes?

works to some degrees but not enough

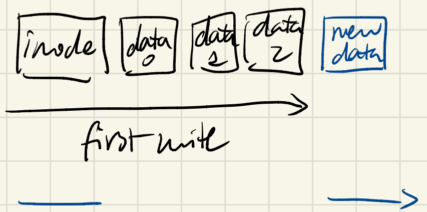
→ try to allocate contiguous blocks whenever possible

☆ delayed allocation : don't allocate blocks for data until we have to write it out, can buffer more writes to allocate a larger contiguous chunk of blocks

→ place frequently updated blocks together

→ inode next to data blocks (does this always work?)

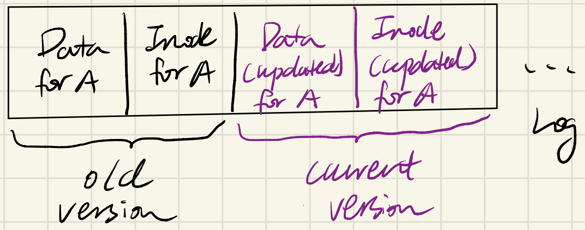
(what about shared bitmaps?)



second write (no longer sequential)

To always perform sequential write ⇒ no fixed location for updates, copy on write!

- → LFS appends changed blocks to a log on disk
- → current metadata & data is the lastest version in the log.



- → to accumulate larger write : buffer enough updates before writing to the log
  - → unit of write = segment

- → metadata & data keeps changing disk loc. upon updates
  - → data can be found via inode
  - → inode? how to find inode? what happens to other fs blocks that point to an inode when an inode moves? (directory)

→ copy on write structures often have recursive update problems!

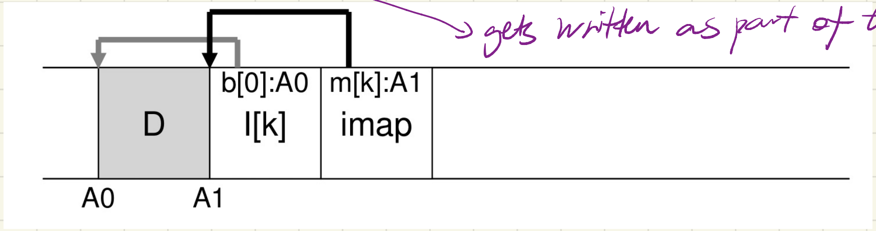
→ can be solved w/ a layer of indirection (similar to SSD FTL indirection)

→ pointer to inode is done via a logical ptr (inode #) instead of its actual location

→ how to translate an inode# ⇒ inode loc?

inode map (many pieces, each tracks a different range of inode #)

→ gets written as part of the update!



accessed frequently, inode map is cached into memory

→ How can we find inode map if it keeps moving?

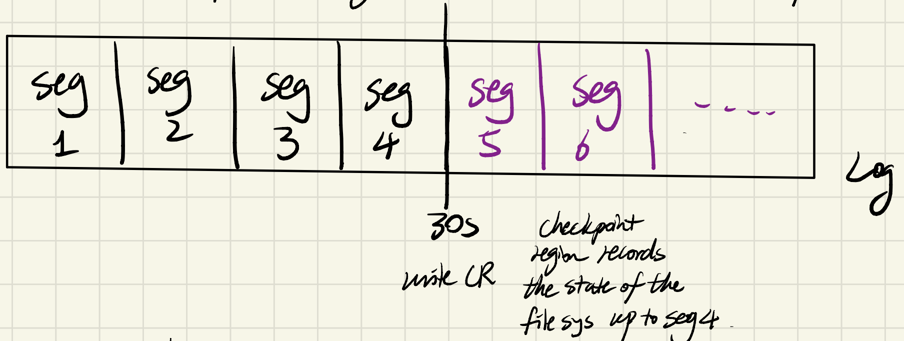
→ loc of each inode map piece is tracked at a fixed location.

LFS tracks ① superblock ② Checkpoint Region at ↗ (written every 30s)

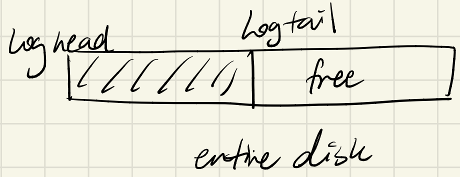
→ segment size
→ FS config

→ loc of inode map pieces
→ last checkpointed segment

→ write to checkpoint region is done at checkpoint interval than segment writes.


seg 1
seg 2
seg 3
seg 4
seg 5
seg 6
.....
Log
30s
write CR
checkpoint region records the state of the file sys up to seg4.

→ Segment Allocation



- no data bitmap, always write sequentially, space btwn tail & head is free
- but what happens when log takes up the entire disk?

★ Garbage Collection of Log

→ segment blocks = some live, some garbage

compact live blocks from multiple segments into a new segment, free to reuse.