Chapter 43: The Log-structured File System (LFS)

1. The Problem with Traditional File Systems

- Traditional file systems (like FFS) perform many small, random writes, which is inefficient on hard drives.
- As memory sizes increase, more data is cached, and disk traffic becomes dominated by writes.

2. The LFS Solution: Sequential Writes

- **Key Idea:** Buffer all updates (data and metadata) in memory and write them to the disk in a single, long, sequential operation called a **segment**.
- LFS never overwrites data in place; it always writes to a new, unused portion of the disk.
- This approach is designed to maximize write performance.

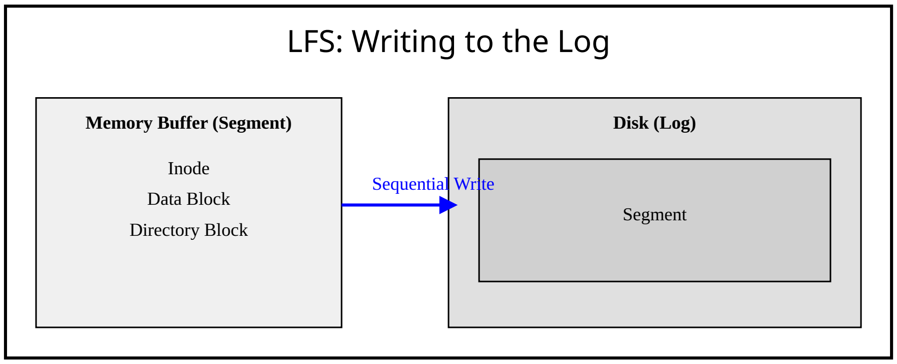

Figure 1: LFS Writing

3. Finding Inodes: The Inode Map (imap)

- **Problem:** Since inodes are written to new locations, LFS needs a way to find them.
- **Solution:** LFS uses an **inode map (imap)**, which is a data structure that maps an inode number to the disk address of the most recent version of that inode.
- The imap itself is also written to the log.
- A **checkpoint region (CR)** at a fixed location on disk points to the latest version of the imap.

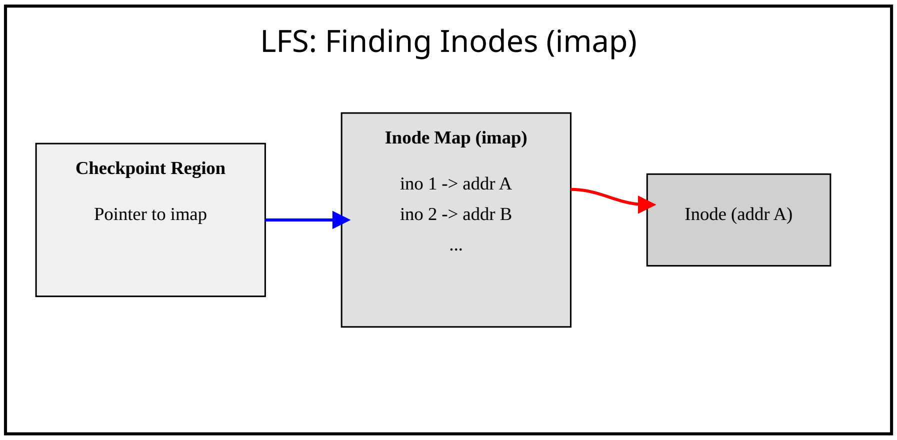

Figure 2: Finding Inodes

4. Garbage Collection (Cleaning)

- **Problem:** Writing new versions of data and metadata to new locations leaves old, invalid versions ("garbage") on the disk.
- **Solution:** LFS uses a **cleaner** process to reclaim this space.
- The cleaner reads old segments, identifies the "live" (still in use) blocks, and writes them to new segments, freeing up the old ones.
- A **segment summary block** helps the cleaner determine which blocks are live.

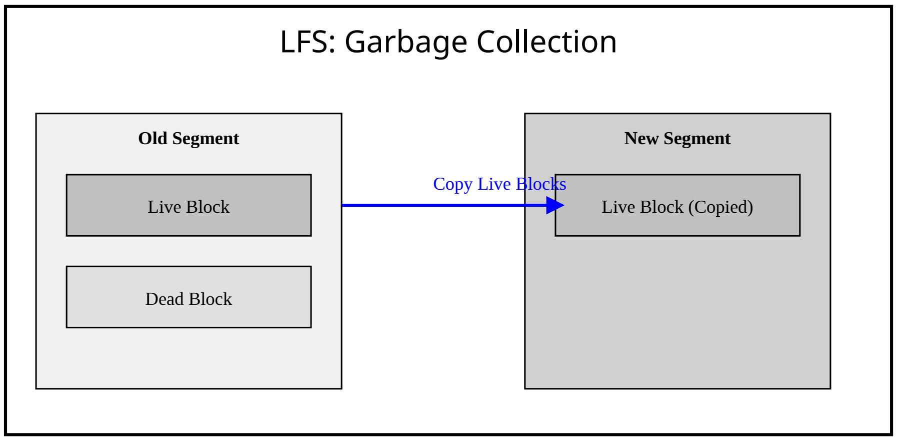

Figure 3: Garbage Collection

5. Crash Recovery

- LFS uses a careful, multi-step write protocol for the checkpoint region to ensure that at least one valid checkpoint is always available.
- After a crash, LFS can recover by reading the last valid checkpoint and then "rolling forward" through the log to apply any subsequent updates.

6. LFS Summary

- LFS is a **copy-on-write** file system that is optimized for write performance.
- **Advantages:**
  - Excellent write performance.
  - Fast recovery from crashes.
- **Disadvantages:**
  - The overhead of garbage collection can be significant.
- The ideas from LFS have influenced many modern file systems, such as ZFS and btrfs.