CS416 – File System
Log-Structured File System

CS 416: Operating Systems Design, Spring 2011

Department of Computer Science
Rutgers University

Rutgers Sakai: 01:198:416 Sp11
(https://sakai.rutgers.edu)

Log Structured Filesystem

Around '91, two trends in disk technology were emerging:

- Disk bandwidth was increasing rapidly (over 40% a year)
- Seek latency not improving much at all
- Machines had increasingly large main memories
    - *Large buffer caches absorb a large fraction of read I/Os*
- Can use for writes as well!
    - *Coalesce several small writes into one larger write*

Some lingering problems with FFS...

- Writing to file metadata (inodes) was required to be synchronous
    - *Couldn't buffer metadata writes in memory*
- Lots of small writes to file metadata means lots of seeks!

LFS Basic Idea

Treat the entire disk as *one big append-only log* for writes!

- Don't try to lay out blocks on disk in some predetermined order
- Whenever a file write occurs, append it to the *end* of the log
- Whenever file metadata changes, append it to the *end* of the log

Collect pending writes in memory and stream out in one big write

- Maximizes disk bandwidth
- No “extra” seeks required (only those to move the end of the log)

When do writes to the actual disk happen?

LFS : Basic Idea

Treat the entire disk as *one big append-only log* for writes!

- Don't try to lay out blocks on disk in some predetermined order
- Whenever a file write occurs, append it to the *end* of the log
- Whenever file metadata changes, append it to the *end* of the log

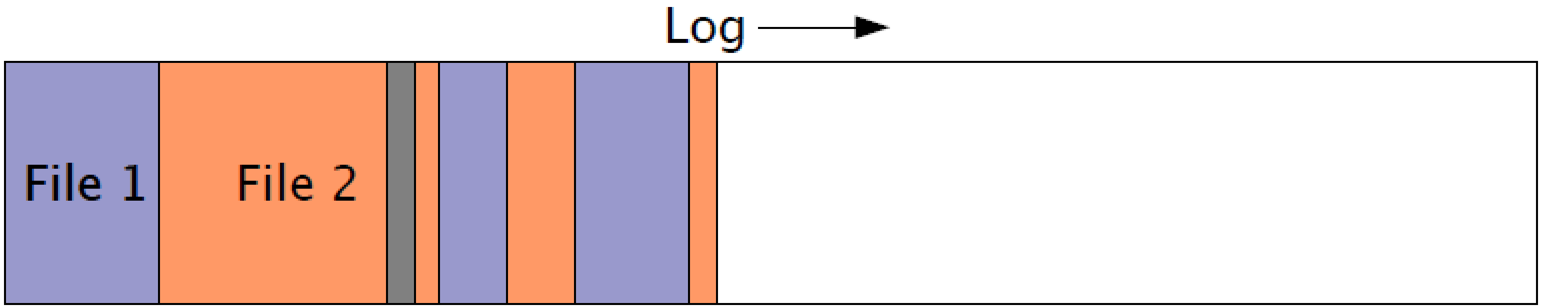
Collect pending writes in memory and stream out in one big write

- Maximizes disk bandwidth
- No “extra” seeks required (only those to move the end of the log)

When do writes to the actual disk happen?

- When a user calls sync() -- synchronize data on disk for whole filesystem
- When a user calls fsync() -- synchronize data on disk for one file
- When OS needs to reclaim dirty buffer cache pages
  - *Note that this can often be avoided, eg., by preferring clean pages*

LFS Example



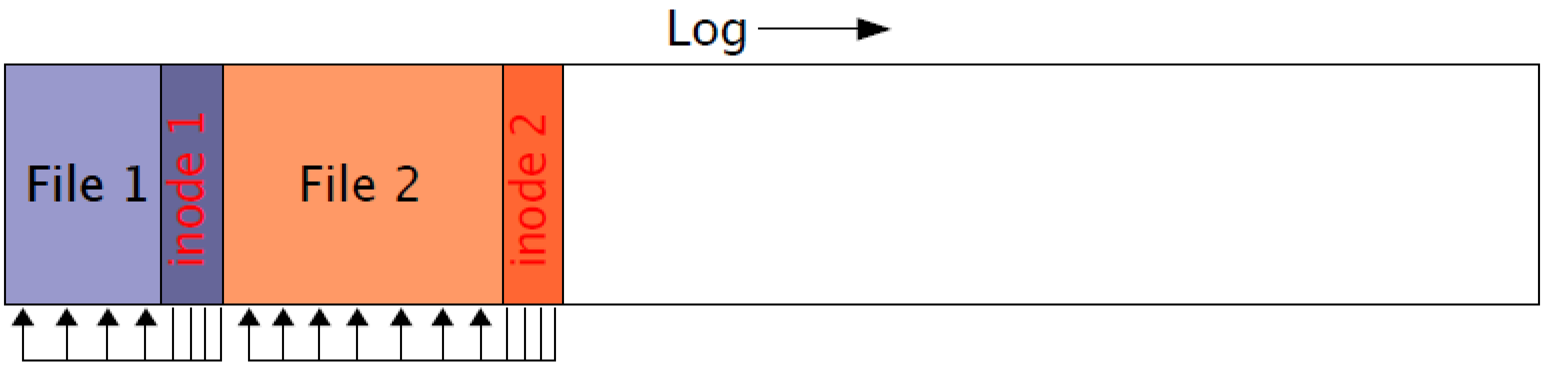
Writing a block in the middle of the file just appends that block to the log

LFS and inodes

How do you locate file data?

- Sequential scan of the log is probably a bad idea …

Solution: Use FFS-style inodes!

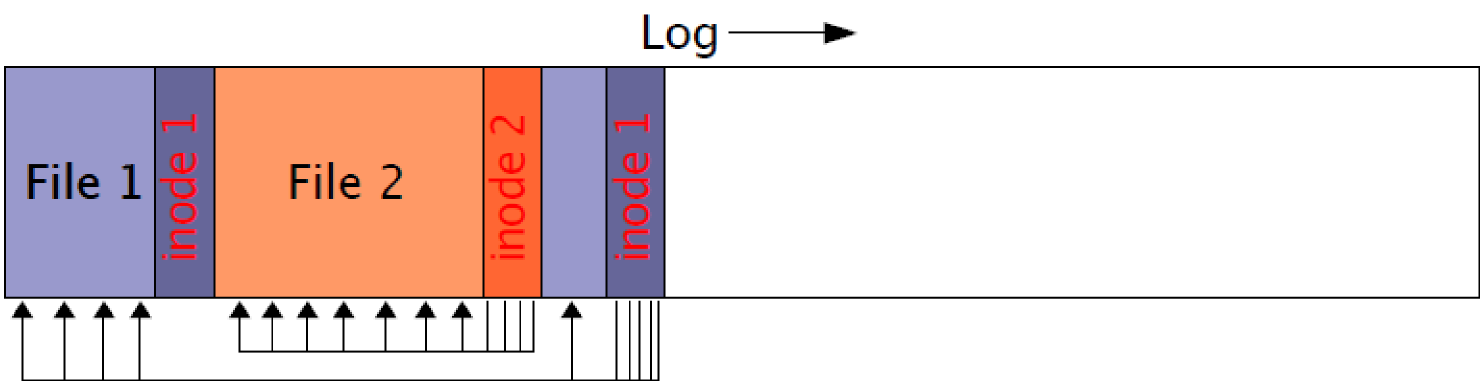


LFS and inodes

How do you locate file data?

- Sequential scan of the log is probably a bad idea …

Solution: Use FFS-style inodes!



Every update to a file writes a new copy of the inode!

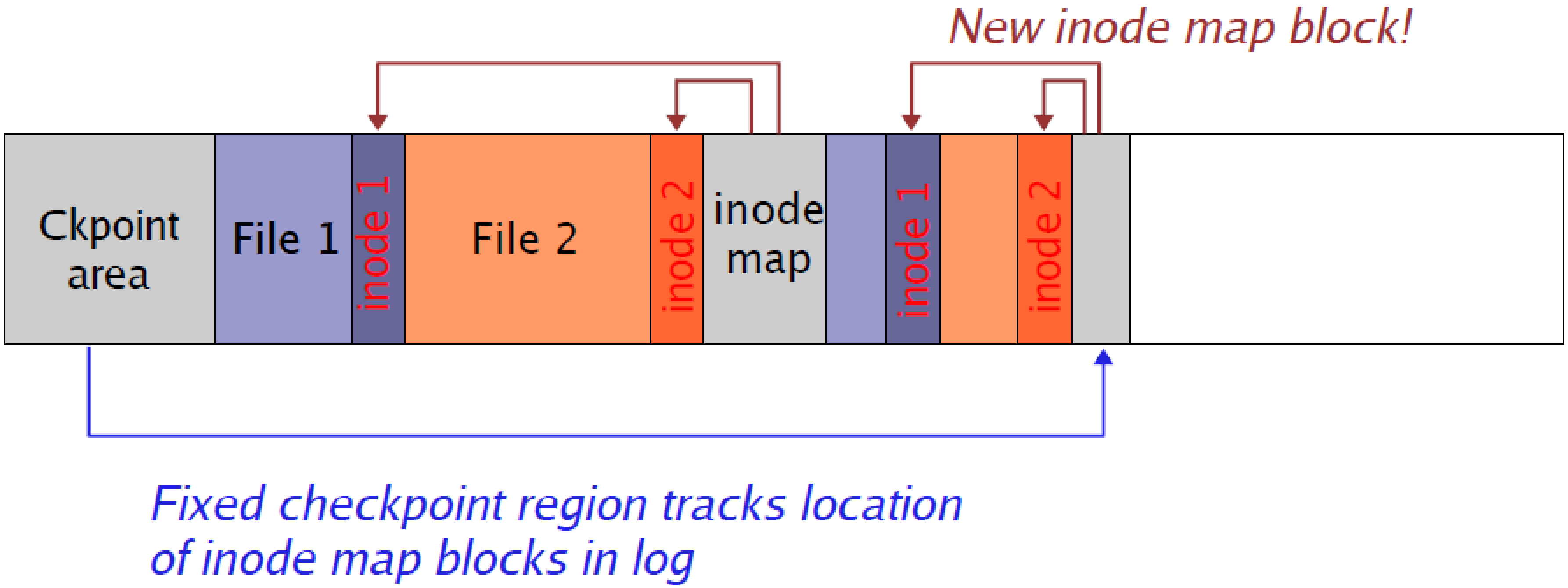
Inode map

Well, now, how do you find the inodes??

- Could also be anywhere in the log!

Solution: *inode maps*

- Maps "file number" to the location of its inode in the log
- Note that inode map is *also* written to the log!!!!
- Cache inode maps in memory for performance



Reading from LFS

But wait ... now file data is scattered all over the disk!

- Seems to obviate all of the benefits of grouping data on common cylinders

Basic assumption: Buffer cache will handle most read traffic

- Or at least, reads will happen to data roughly in the order in which it was written
- Take advantage of huge system memories to cache the heck out of the FS!

Log Cleaner

With LFS, eventually the disk will fill up!

- Need some way to reclaim “dead space”

What constitutes “dead space?”

- Deleted files
- File blocks that have been “overwritten”

Solution: Periodic “log cleaning”

Scan the log and look for deleted or overwritten blocks

- Effectively, clear out stale log entries

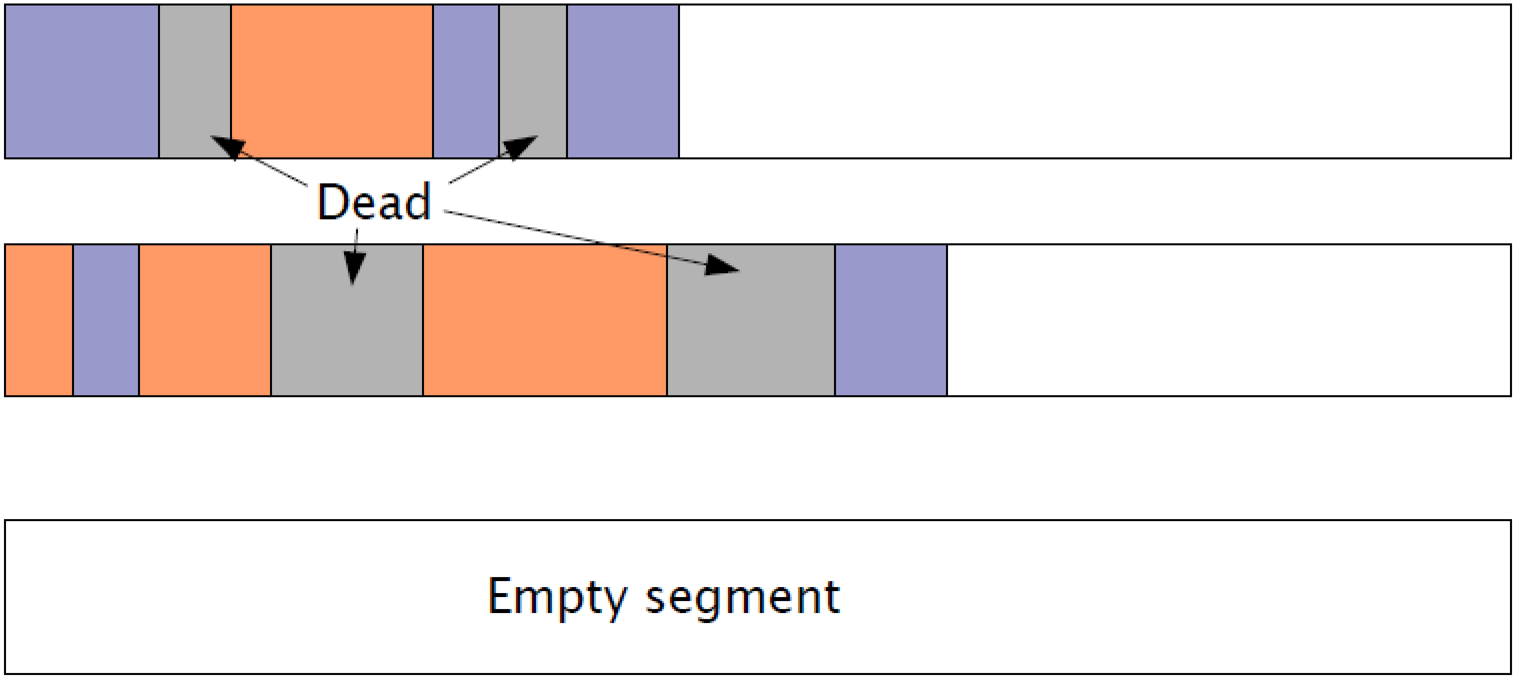
Copy *live data* to the end of the log

- The rest of the log (at the beginning) can now be reused!

Log Cleaning Example

LFS cleaner breaks log into *segments*

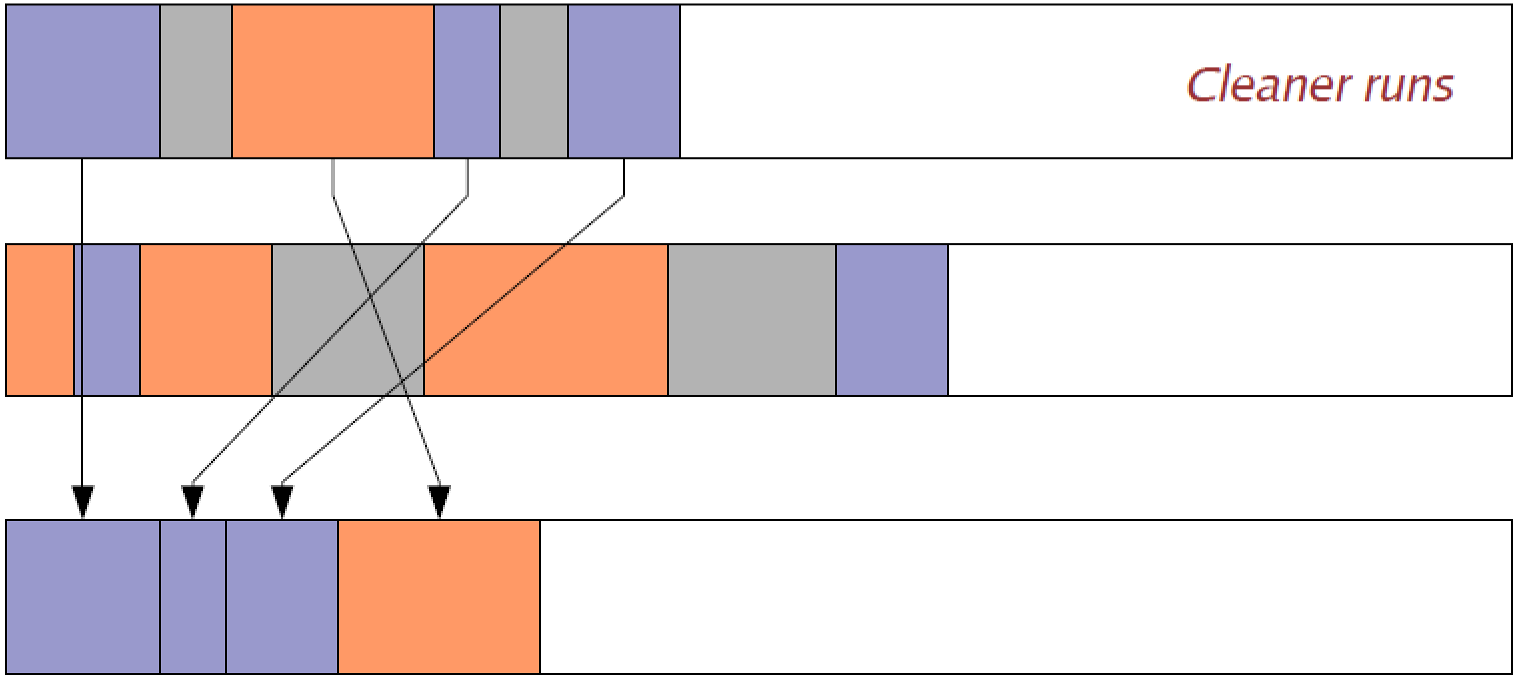
- Each segment is scanned by the cleaner
- Live blocks from a segment are copied into a new segment
- The entire scanned segment can then be reclaimed



LFS Cleaning Example

LFS cleaner breaks log into *segments*

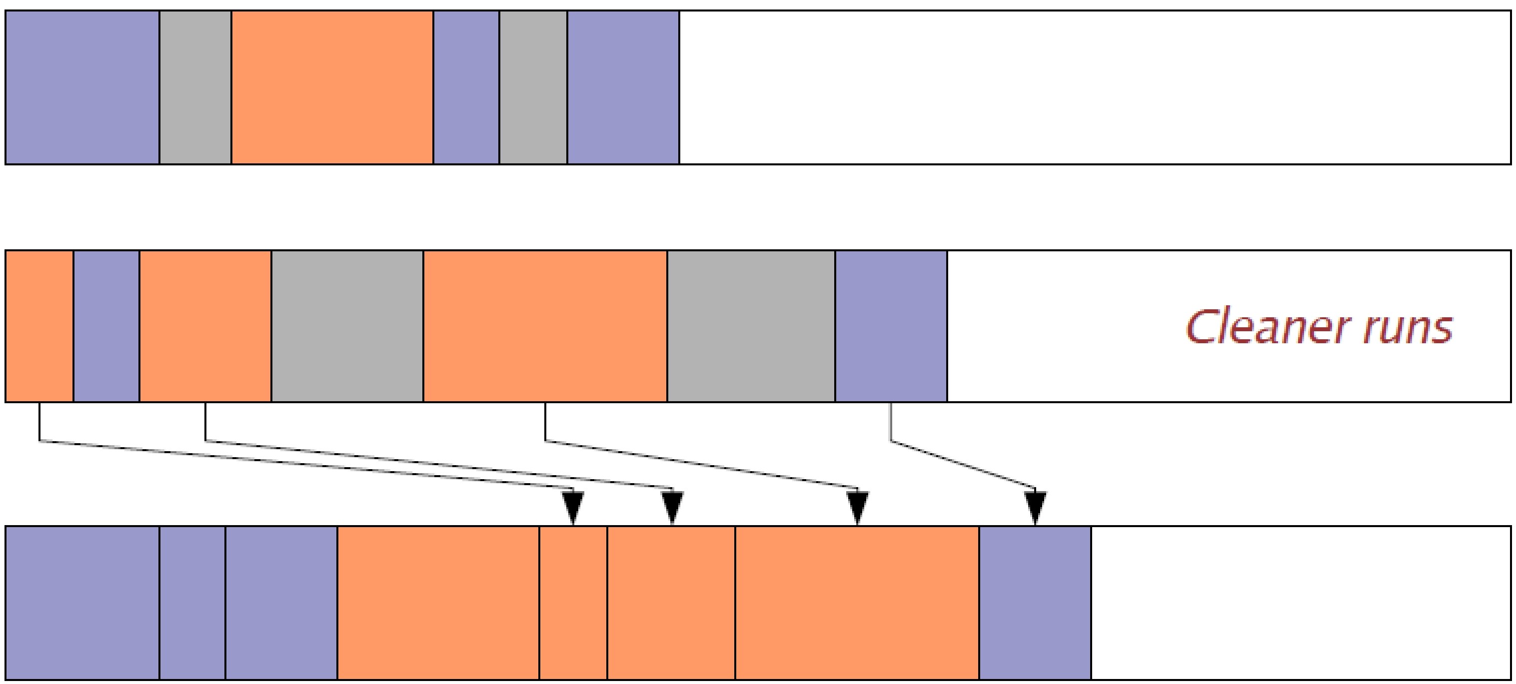
- Each segment is scanned by the cleaner
- Live blocks from a segment are copied into a new segment
- The entire scanned segment can then be reclaimed



LFS Cleaning Example

LFS cleaner breaks log into *segments*

- Each segment is scanned by the cleaner
- Live blocks from a segment are copied into a new segment
- The entire scanned segment can then be reclaimed



LFS Cleaning Example

LFS cleaner breaks log into *segments*

- Each segment is scanned by the cleaner
- Live blocks from a segment are copied into a new segment
- The entire scanned segment can then be reclaimed

Cleaning issues

When does the cleaner run?

- Generally when the system (or at least the disk) is otherwise idle
- Can cause problems on a busy system with little idle time

Cleaning a segment requires reading the whole thing!

- Can reduce this cost if the data to be written is already in cache