10/28 CS240 - LFS

Announcements

For next class (Thursday 10/30)

1. Read: Rethink the Sync
2. Submit answers to reading questions (see course schedule) before class

Looking ahead:

Lab 1 is due end of day November 2nd (**this Sunday**)

Paper backgrounds

- LFS - 1991 Symposium on Operating Systems Principles (SOSP)
  - Award paper forwarded to TOCS
  - 2014 IEEE Reynold B. Johnson Information Storage Systems Award

Disk storage management review

- Small file create modifications
    - file inode, file block, directory block, directory inode, free bitmap block
- BSD Fast File System
    - Directory locality - cylinder groups
    - Sequential layout of files - free bitmap data structure
    - Crash recovery
        - Synchronous writes of most metadata
        - fsck - metadata, bitmap consistency restore
- Write-ahead logging - write twice: first to log, delayed to block location
    - Write performance
    - Crash recovery

LFS Motivation

- Not in paper: RAID
- Technology trend bet: CPU faster, RAM bigger, disk still mechanical
- Change in Read/Write ratio seen by storage
    - Big RAM file caches work well for reads but less so for writes
    - NFS stateless?
- Small-file workloads are interesting

Log-structured file systems

- Make all writes sequential, like a log, but support reading back from the log
- Challenges:
    - Most files system place file descriptors in known locations on disk (inum -> diskaddr is easy)
        - File systems metadata locations moves when written in LFS
    - What happens when your log wraps around?
        - Threading
        - Write live blocks back to the log
        - Free space compaction
    - How do you evaluate something that will degrade some over time?
        - How about file system requiring a disk defragmenter run periodically?

What is read when an LFS file system is mounted?

- Fetch root directory: inode 2
- What is different from Unix FFS?

Segment cleaning

- Read N segments, write out N-1 or fewer segments to the log, mark N free
- Given a disk address, how can I tell what it belongs to?
  - Segment summary blocks
  - Even harder in LFS?
  - Why generation number in Inode map? Is the NFS generation number the same?
- Why might partial segment writes be common?
  - Why are multiple segment summary blocks needed?

Announcements

Lab 1 is due end of day November 2nd (this Sunday)

No class next Tuesday (11/4): Stanford holiday - Democracy Day

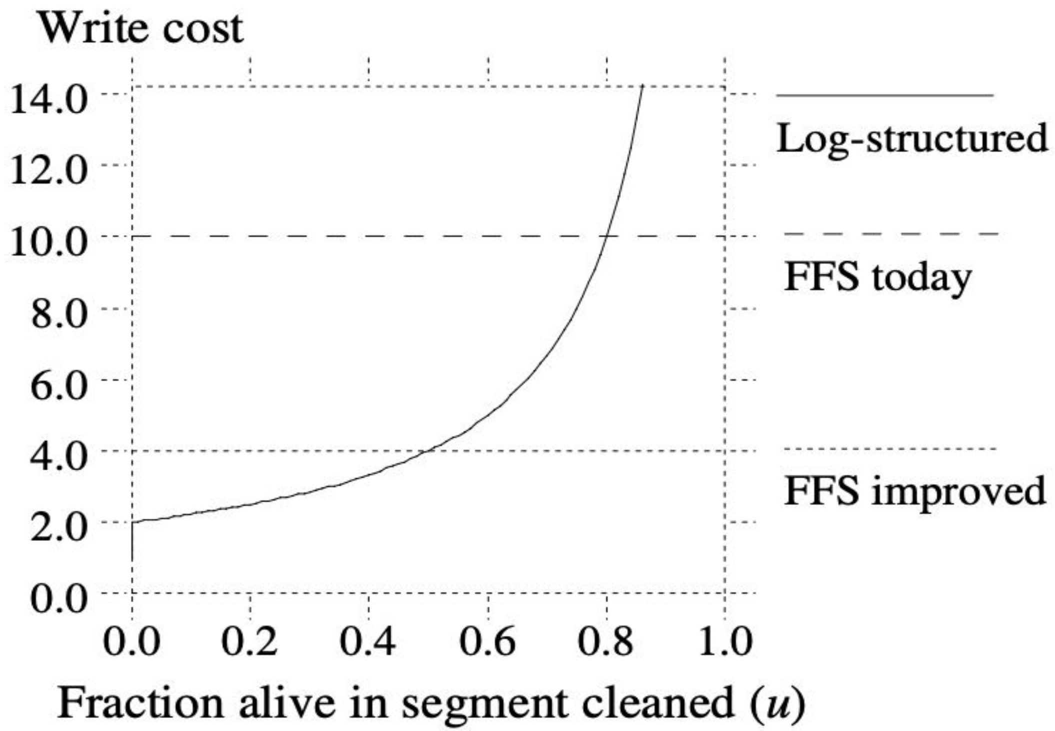
For next class (Thursday 11/6)

1. Read: F2FS: A New File System for Flash Storage
2. No reading questions

Interesting questions

- Simulation technique useful for exploring space
- Which segments to read?
    - Greedy approach not the best
- How to organize the live data written?
    - Reorganizing data for read performance not the best

What is the write cost metric? What is 1?



Simulations with hot-and-cold files

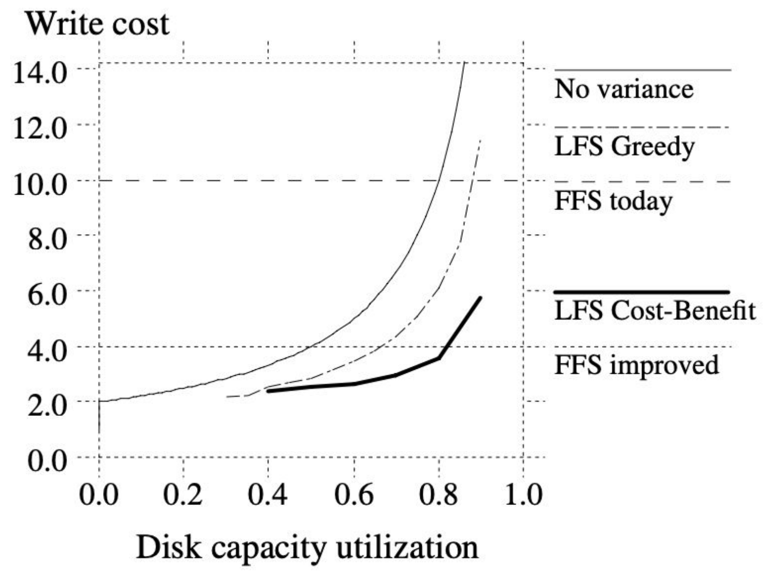
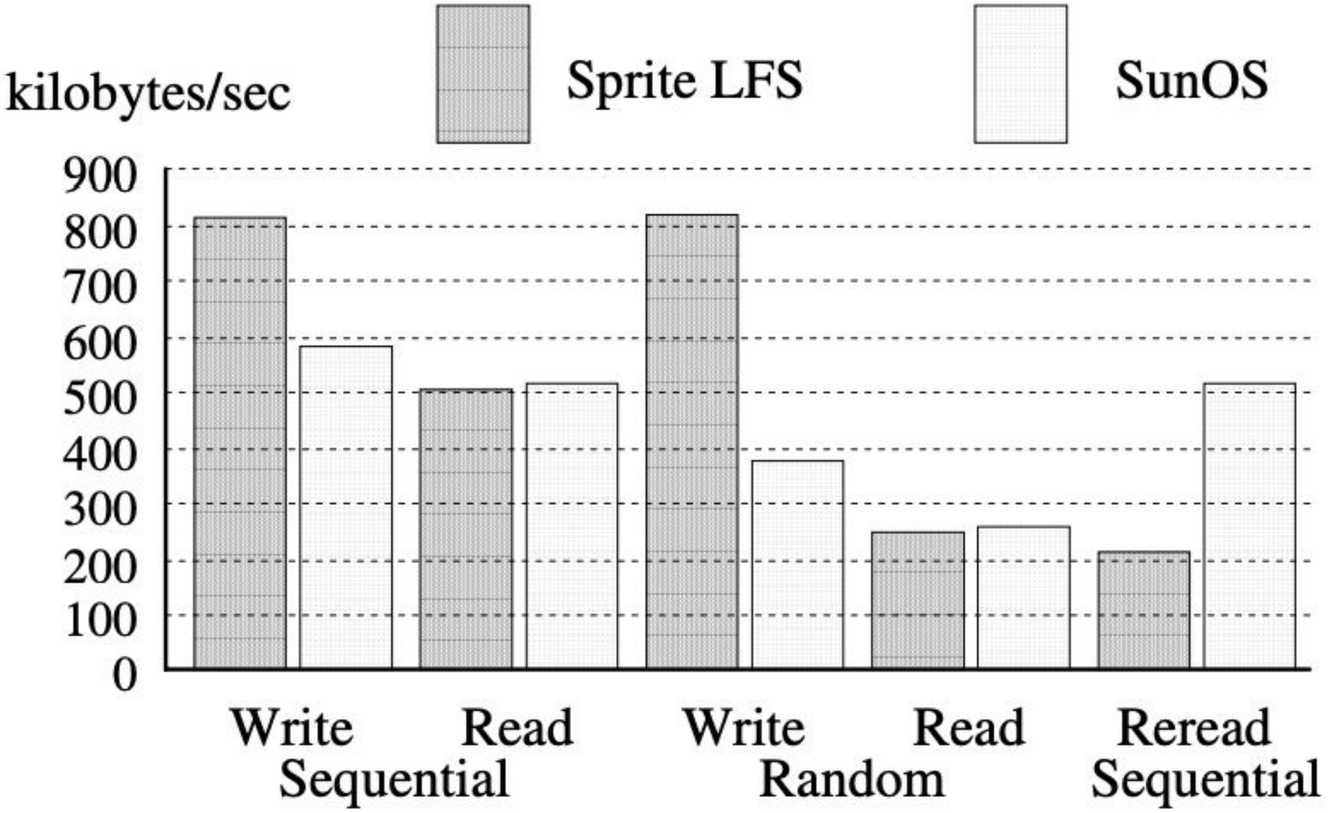


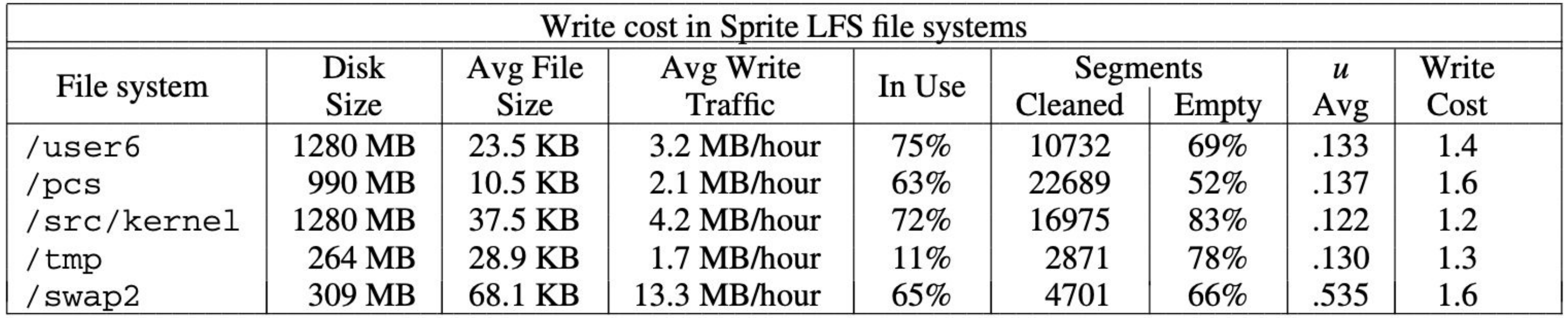
Figure 7 — Write cost, including cost-benefit policy.
This graph compares the write cost of the greedy policy with that of the cost-benefit policy for the hot-and-cold access pattern. The cost-benefit policy is substantially better than the greedy policy, particularly for disk capacity utilizations above 60%.

Crash recovery - Roll forward

- How did LFS find the log?
- segment summary block
- directory operation log?


kilobytes/sec
Sprite LFS
SunOS
900
800
700
600
500
400
300
200
100
0
Write
Read
Sequential
Write
Read
Random
Reread
Sequential

Real system measurements



Write cost in Sprite LFS file systems

File system	Disk Size	Avg File Size	Avg Write Traffic	In Use	Segments		u	Write
					Cleaned	Empty	Avg	Cost
`/user6`	1280 MB	23.5 KB	3.2 MB/hour	75%	10732	69%	.133	1.4
`/pcs`	990 MB	10.5 KB	2.1 MB/hour	63%	22689	52%	.137	1.6
`/src/kernel`	1280 MB	37.5 KB	4.2 MB/hour	72%	16975	83%	.122	1.2
`/tmp`	264 MB	28.9 KB	1.7 MB/hour	11%	2871	78%	.130	1.3
`/swap2`	309 MB	68.1 KB	13.3 MB/hour	65%	4701	66%	.535	1.6

Table 2 - Segment cleaning statistics and write costs for production file systems.
For each Sprite LFS file system the table lists the disk size, the average file size, the average daily write traffic rate, the average disk capacity utilization, the total number of segments cleaned over a four-month period, the fraction of the segments that were empty when cleaned, the average utilization of the non-empty segments that were cleaned, and the overall write cost for the period of the measurements. These write cost figures imply that the cleaning overhead limits the long-term write performance to about 70% of the maximum sequential write bandwidth.

Data from real system

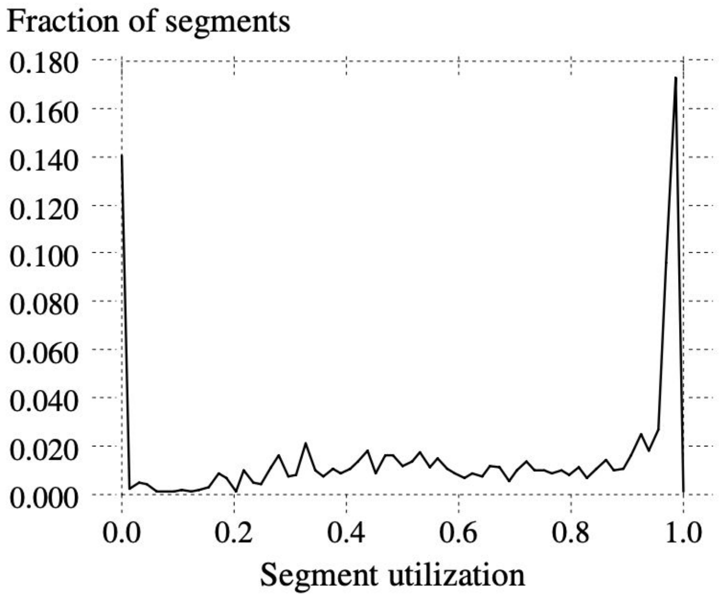


Figure 10 — Segment utilization in the /user6 file system
This figure shows the distribution of segment utilizations in a recent snapshot of the `/user6` disk. The distribution shows large numbers of fully utilized segments and totally empty segments.

Sprite LFS `/user6` file system contents		
Block type	Live data	Log bandwidth
Data blocks*	98.0%	85.2%
Indirect blocks*	1.0%	1.6%
Inode blocks*	0.2%	2.7%
Inode map	0.2%	7.8%
Seg Usage map*	0.0%	2.1%
Summary blocks	0.6%	0.5%
Dir Op Log	0.0%	0.1%