11/5 CS240 - F2FS

Announcements

For next class (Tuesday 11/10)

1. Read: Application Performance and Flexibility on Exokernel Systems
2. No reading questions

Paper

- F2FS: A New File System for Flash Storage
  - FAST 2015 - Proceedings of the 13th USENIX Conference on File and Storage Technologies
- F2FS is an important flash-optimized file system on Android and embedded Linux

Background: I/O Performance Gap in year 2007

- DRAM main memory
  - Access time: ~60–80 ns
  - Typical Capacity: 1 GB – 8 GB
  - Cost: $50–$100 / GB
  - Volatile
- Magnetic disks
  - Access time: ~8–12 ms
  - Typical Capacity: 150 – 1000 GB
  - Cost: $0.25–$0.40 / GB
  - Non Volatile

- Disk <-> Memory performance gap 5 orders of magnitude, but non volatile, 200x cheaper
  - Access time: disk ~125,000× slower
  - Bandwidth: disk ~75× slower
  - IOPS: disk ~100,000× slower

- Many new solid-state (not mechanical) technologies proposed to address gap
  - Flash got a push from consumer devices: digital cameras, music players, mobile
  - Flash read performance was good, writes and write endurance a challenge

Flash memory chip characteristics

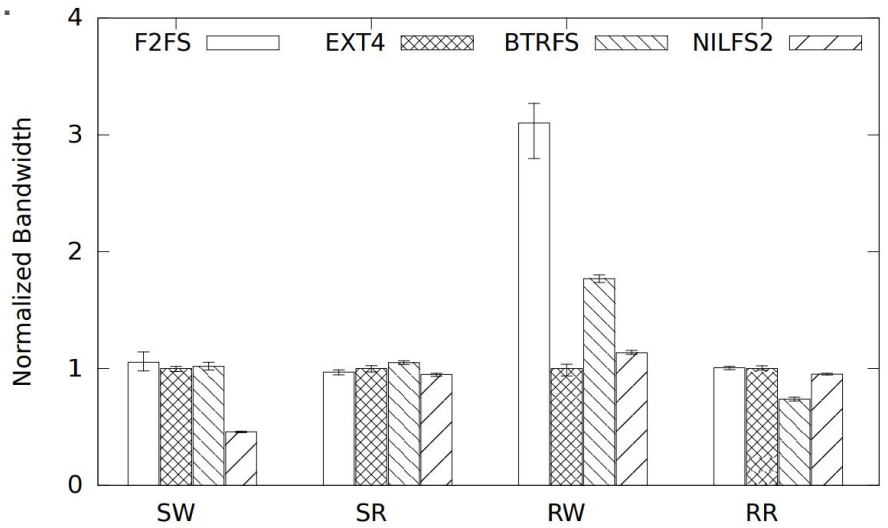
- Organized as a linear array of pages (2-4KB)
  - Same as magnetic disks used, good match for an OS file system
  - Read access times (50–100 µs) much closer to memory than disk
- Writes are a problem
  - Write time (200–1000 µs) 5-10x slower than reads
  - Can not update a page in place, need to "erase" previous value first
  - Erase only works on a "block" contains 128–1024 pages
    - Erase is slow 2-10ms (~ same disk seek time), 200–1000× slower than reads
    - Erase wears out block: Errors creep in after 3000–10000 erases
- New file systems were needed to handle this write weirdness
  - Could go back to Flash memory vendors and tell them to make it work like a disk
  - Which do you think happened?

Flash Translation Layer (FTL) - Make Flash like Disk

- Maintains a mapping table
  - Logical Page Number (LPN) ⇒ Physical Page Number (PPN)
  - Challenge: 1 TB SSD / 2 KB pages ⇒ ~512 M entries (2GB too large for fast memory)
    - Logical block number (LBN) ⇒ physical block number (PBN).
- Hides erase-before-write constraint
  - Write new data to a fresh page and remapping the logical address
  - Challenge: Having to copy entire logical block too much overhead
    - Take a page for LFS: log blocks
      - Direct, Set-associative (assumed in paper, why good?), Fully-associative
      - Choosing blocks (greedy vs cost-benf) to "clean" by merging log block, must erase before reuse
        - Added "trim" (discard) command for OS to inform FTL logical blocks invalid
- Performs *wear leveling* by rotating which physical blocks get used
  - Tracks erase cycles per block
- Random writes cause *write amplification*: a problem for FTL
  - Why is sequential writes less a problem?
- Why might you think a file system could handle this better?

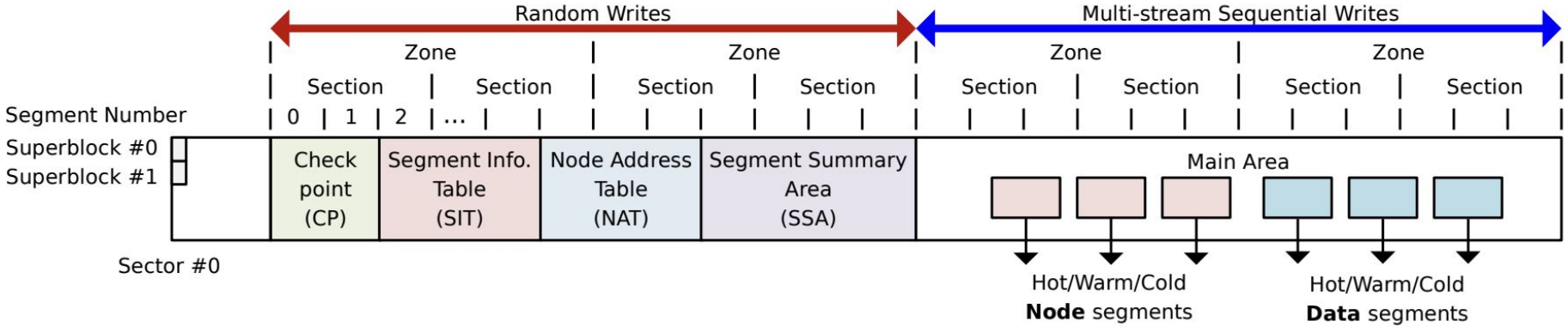
Is LFS a good file system for FTL (Log on Log)?

- Large, sequential writes are good, eliminate random writes
- Other properties less good of fit:
  - Write amplification
  - "Wandering tree" problem
- NI**LFS**2 didn't do so well:
  - NILFS2 good at checkpoints
- F2FS tried to do better
  - Incorporated other known good ideas



(a) iozone

F2FS logical address layout



- What is the function of:
  - Checkpoint (CP)
  - Segment info Table (SIT)
  - Node Address Table (NAT)
  - Segment Summary Area (SSA)
- What are segments, sections, and zones?
- Why separate Node and Data?
- Why Hot/Warm/Cold logs?

Why?

Table 1: Separation of objects in multiple active segments.

Type	Temp.	Objects
Node	Hot	Direct node blocks for directories
	Warm	Direct node blocks for regular files
	Cold	Indirect node blocks
Data	Hot	Directory entry blocks
	Warm	Data blocks made by users
	Cold	Data blocks moved by cleaning; Cold data blocks specified by users; Multimedia file data

Wandering tree problem and solution?

- How does Node Address Table compare to LFS's inode Map?
  - Where is it written on disk?
- Why did the make the inode 4KB?
- How to avoid write amplification when modifying NAT?
- What is inline data?
- What is the mistake in Sec 2.2:

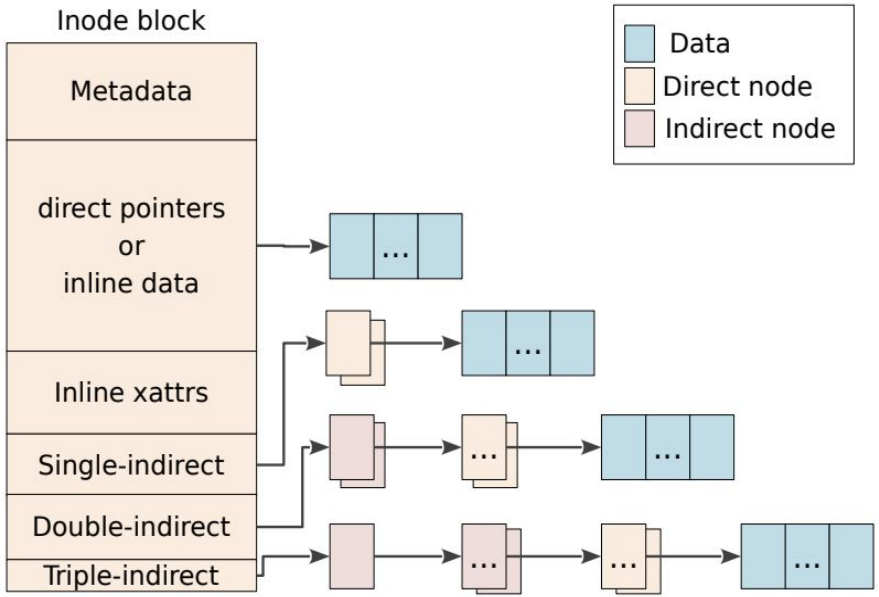


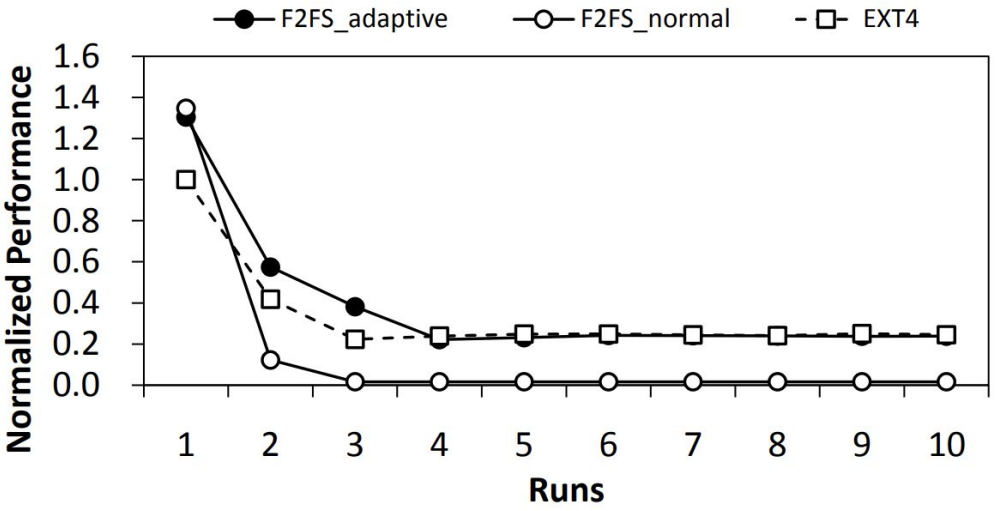
Figure 2: File structure of F2FS.

types: inode, direct and indirect node. An inode block contains a file's metadata, such as file name, inode number, file size, atime and dtime. A direct node block con-

How is the directory structure different than LFS?

How do you look up "/dir/file" contents?

What is adaptive logging?



(b) iozone (random updates) on a device filled up 100%.

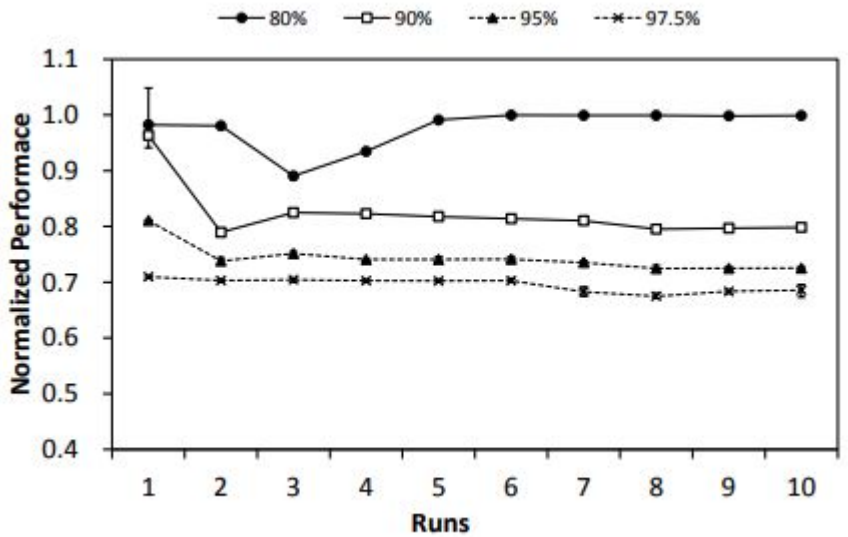
Crash Recovery?

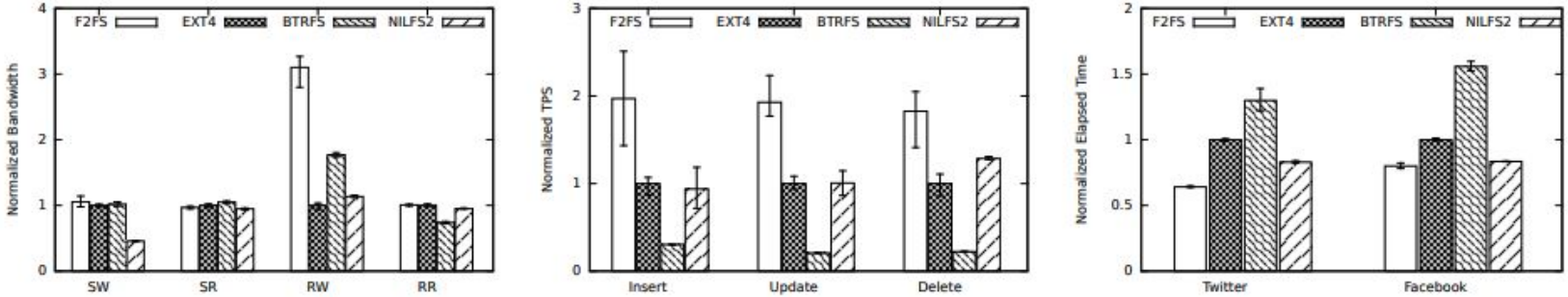
- What happens in a checkpoint?
- What are orphan inodes?

What do you have to write after fsync?

How does cleaning work?

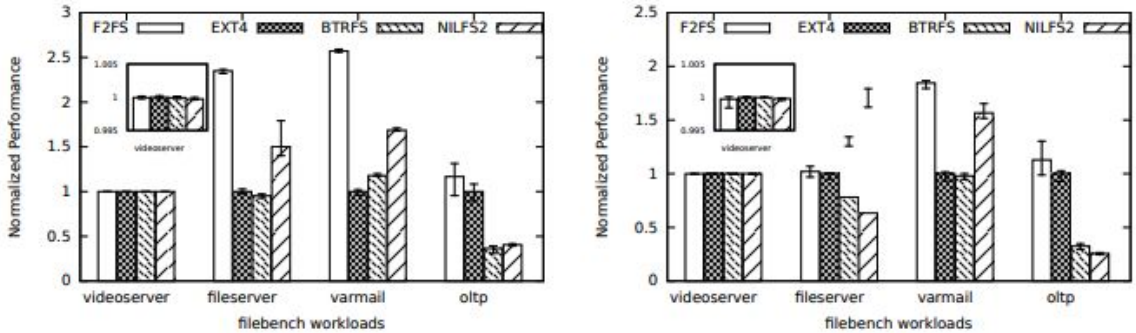
- Foreground? Background? How different?
- How is Background implemented?





(a) iozone (b) SQLite (c) Facebook-app and Twitter-app

Figure 3: Performance results on the mobile system.



(a) SATA SSD (b) PCIe SSD

Figure 4: Performance results on the server system.