Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

Seoul National University

Fall 2020

F2FS

(M. Rosenblum and J. K. Ousterhout., SOSP, 1991)

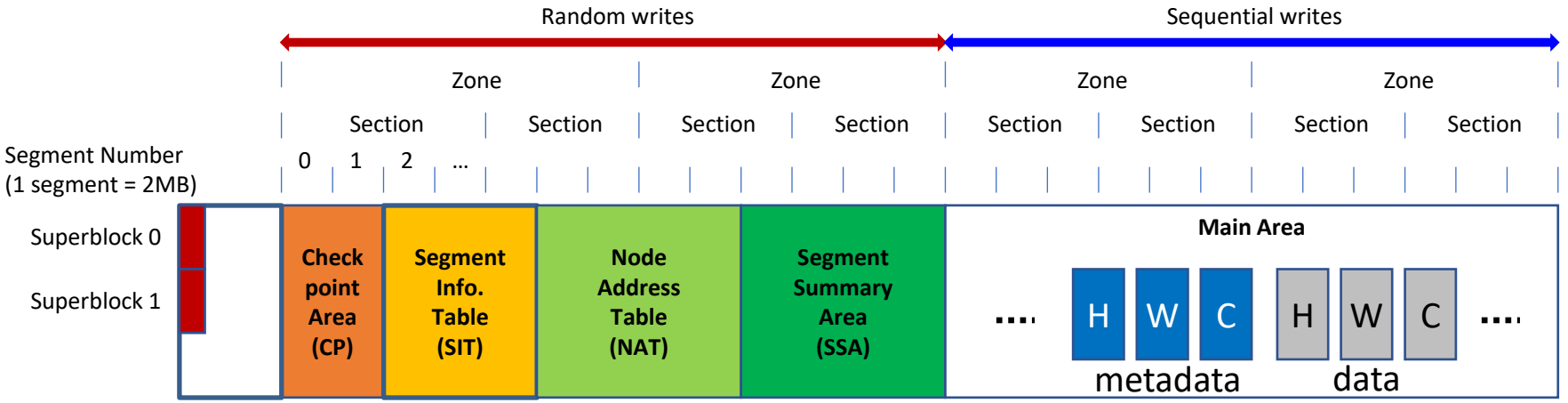
F2FS

- Based on the Log-structured File System (LFS)
- Flash-friendly on-disk layout
- Cost-effective index structure
- Multi-head logging
- Adaptive logging
- Fsync acceleration with roll-forward recovery

- Publicly available, included in Linux mainline kernel since Linux 3.8
- Being used in commercial Android smartphones

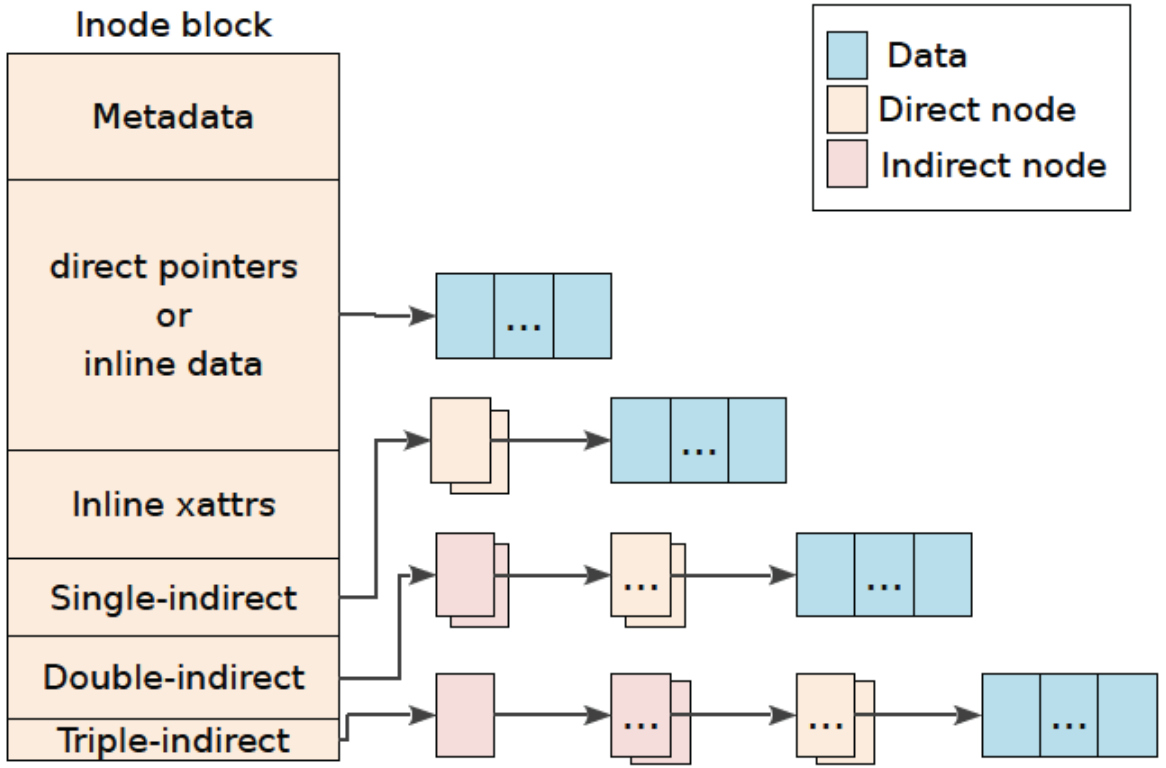
Flash-friendly On-Disk Layout

- Flash awareness
  - All the metadata are located together for locality
  - Start address of main area is aligned to the zone size
  - Cleaning is done in a unit of section (FTL's GC unit)
- Cleaning cost reduction
  - Multi-head logging for hot/cold data separation

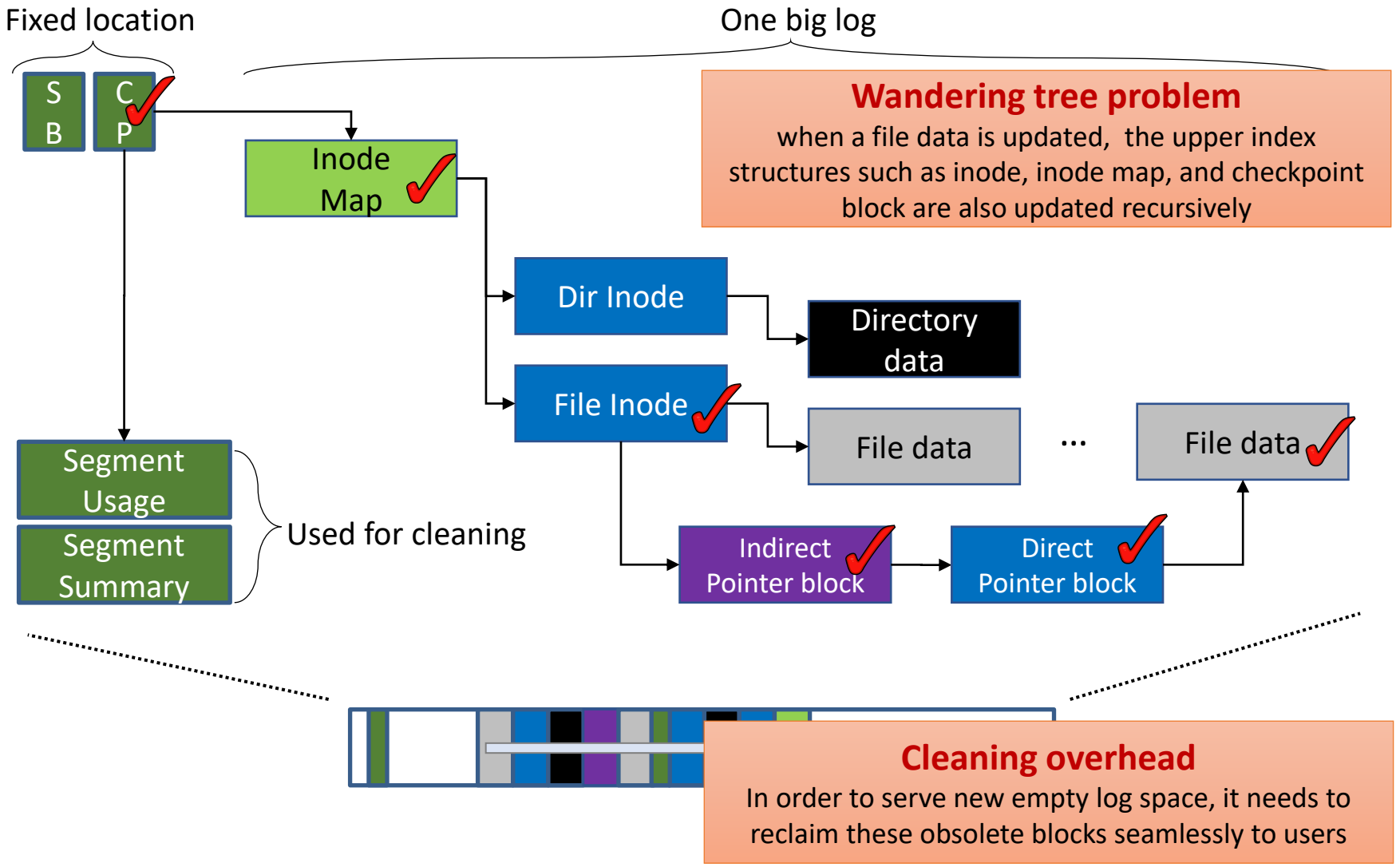


Inode

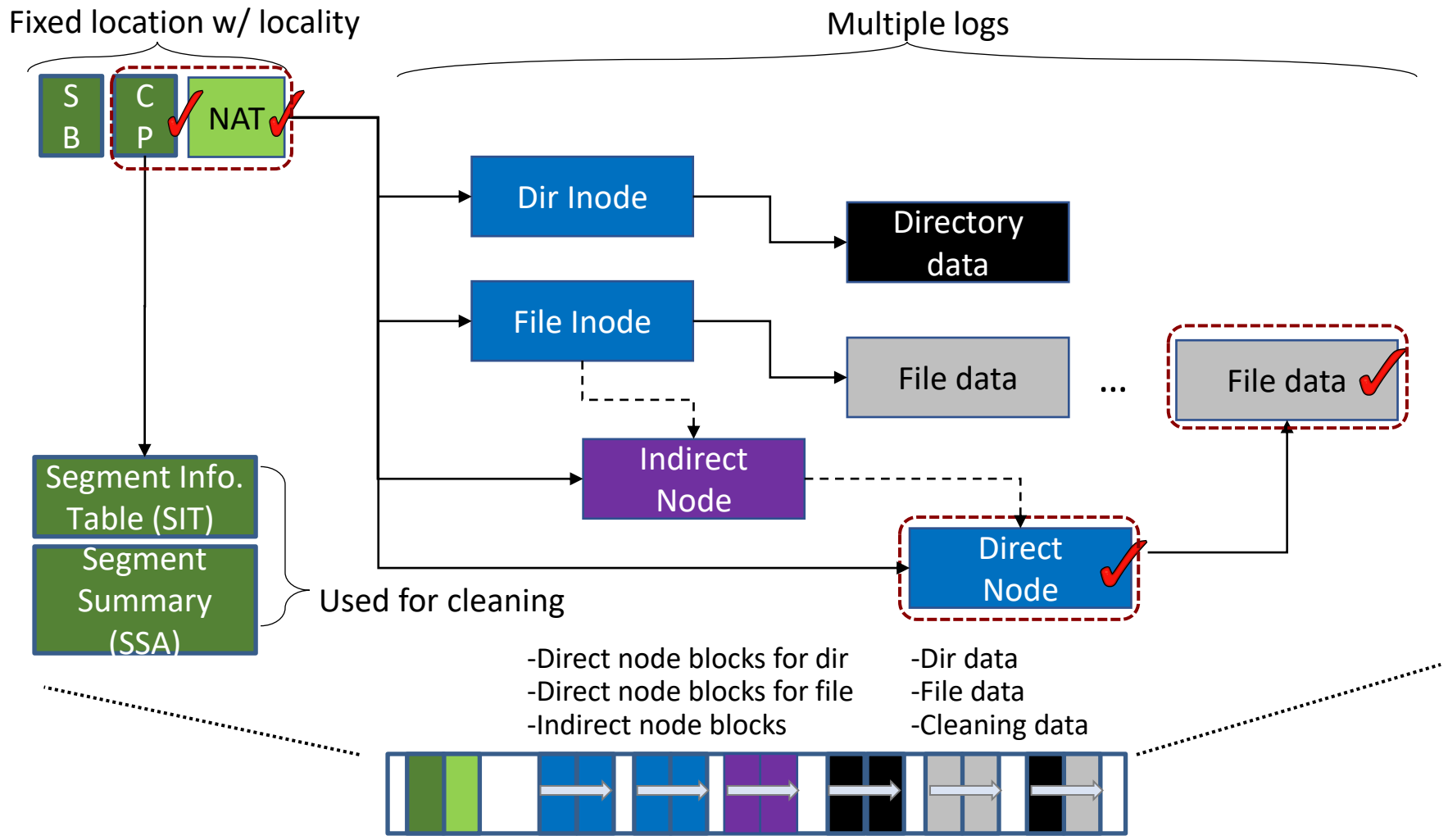
- Inline data up to 3,692 bytes
- Inline extended attributes up to 200 bytes



LFS Index Structure



F2FS Index Structure



Multi-head Logging

- Data temperature classification
  - Node > Data
  - Direct node > Indirect node
  - Directory > User file

- Separation of multi-head logs in NAND flash
  - Zone-aware log allocation for set-associative mapping FTL
  - Multi-stream interface

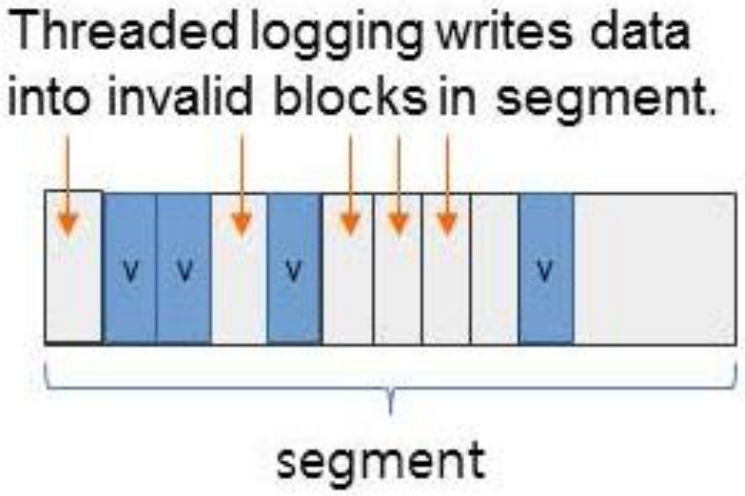
Type	Temp.	Objects
Node	Hot	Direct node blocks for directories
	Warm	Direct node blocks for regular files
	Cold	Indirect node blocks
Data	Hot	Directory entry blocks
	Warm	Data blocks made by users
	Cold	Data blocks moved by cleaning; Cold data blocks specified by users; Multimedia file data

Adaptive Logging

- To reduce cleaning cost at highly aged conditions, F2FS changes write policy dynamically

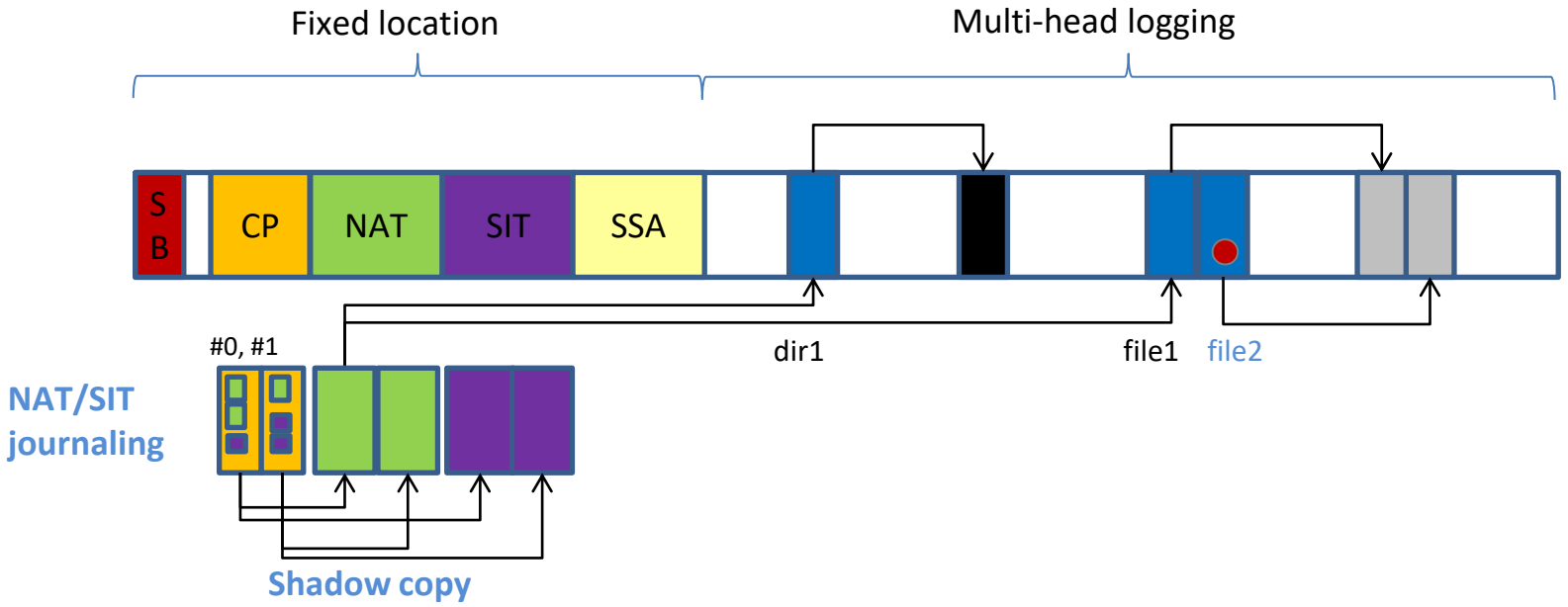
- Append logging (logging to clean segments)
  - Need cleaning operations if there is no free segment
  - Cleaning causes mostly random read and sequential writes

- Threaded logging (logging to dirty segments)
  - Reuse invalid blocks in dirty segments
  - No cleaning needed
  - Cause random writes



Sudden Power Off Recovery

- Checkpoint and rollback
  - Maintains shadow copy of checkpoint, NAT, SIT blocks
  - Recovers the latest checkpoint
  - Keeps NAT/SIT journal in checkpoint to avoid NAT, SIT writes



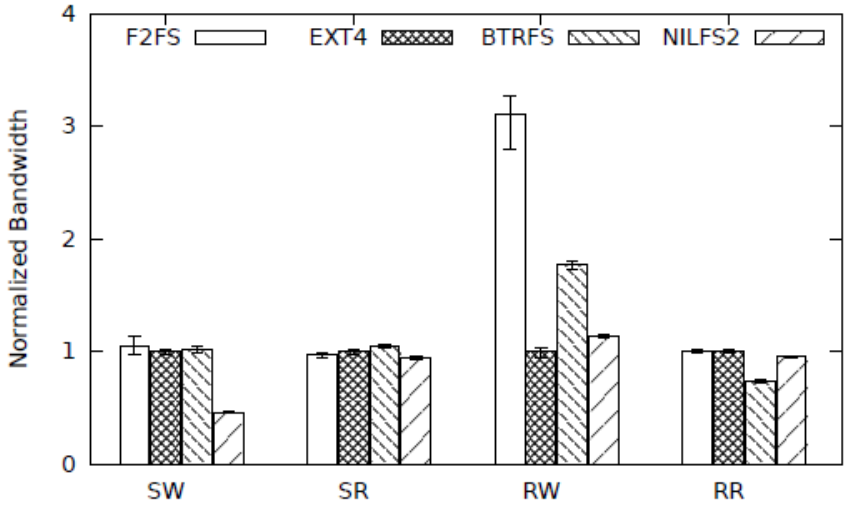
Fsync Acceleration

- Fsync handling
  - On fsync, checkpoint is not necessary
  - Direct node blocks are written with fsync mark

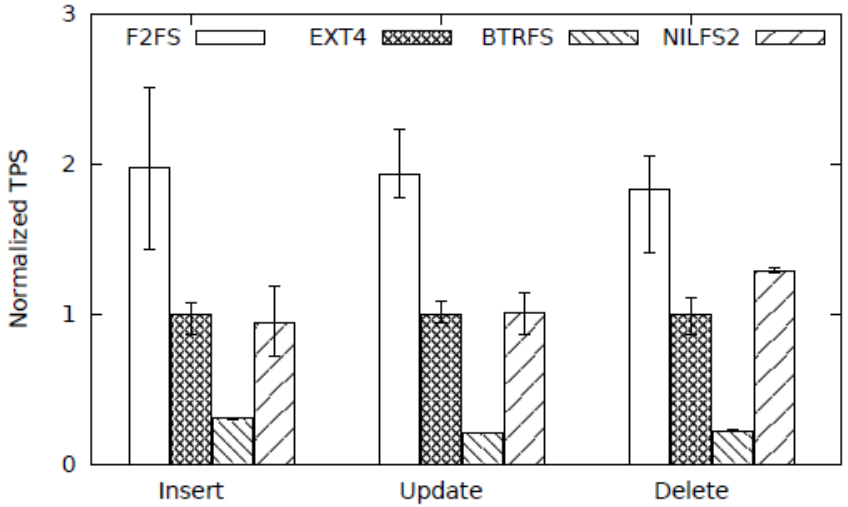
- Roll-forward recovery procedure
  - Search marked direct node blocks
  - Per marked node block, identify old and new data blocks by checking the difference between the current and previous node block
  - Update SIT; invalidate old data blocks
  - Replay new data block writes; update NAT, SIT accordingly
  - Create checkpoint

Mobile Benchmark

- In F2FS, more than 90% of writes are sequential
- F2FS reduces write amount per fsync by using roll-forward recovery
- Btrfs and Nilfs2 performed poor than Ext4
  - Btrfs: heavy indexing overheads
  - Nilfs2: periodic data flush



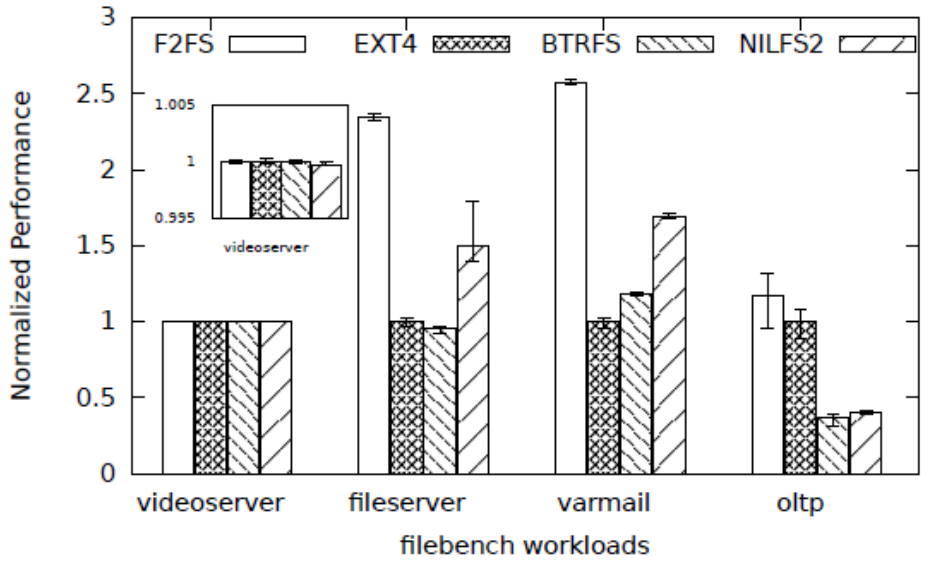
(a) iozone



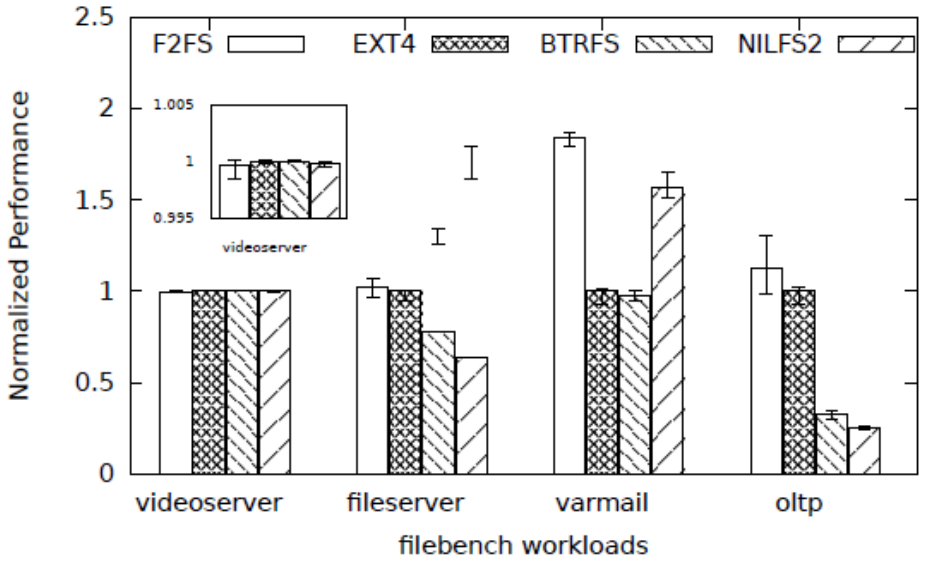
(b) SQLite

Server Benchmark

- Performance gain of F2FS over Ext4 is more on SATA SSD than PCIe SSD
- Discard size matters in SATA SSD due to interface overhead

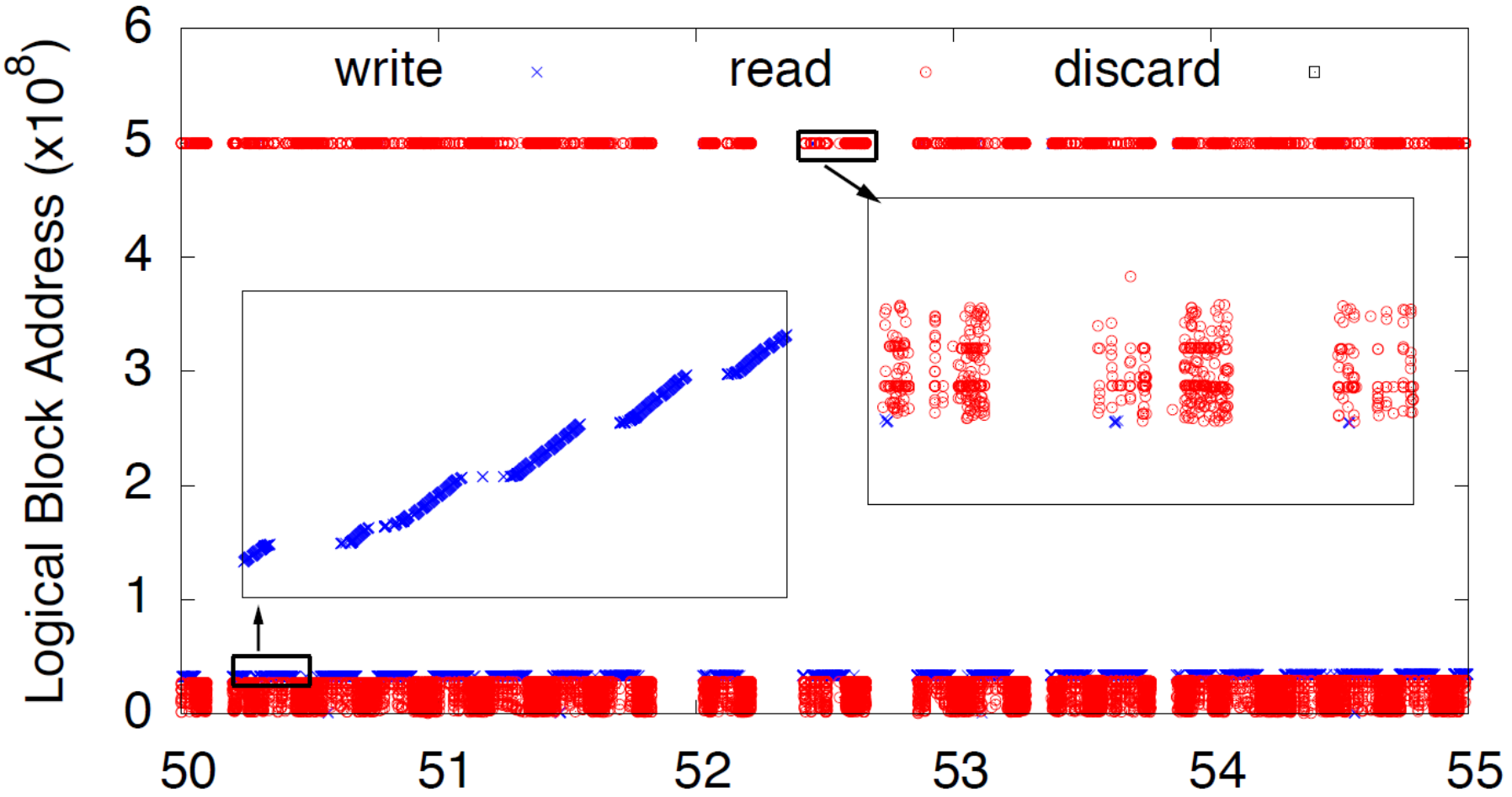


(a) SATA SSD

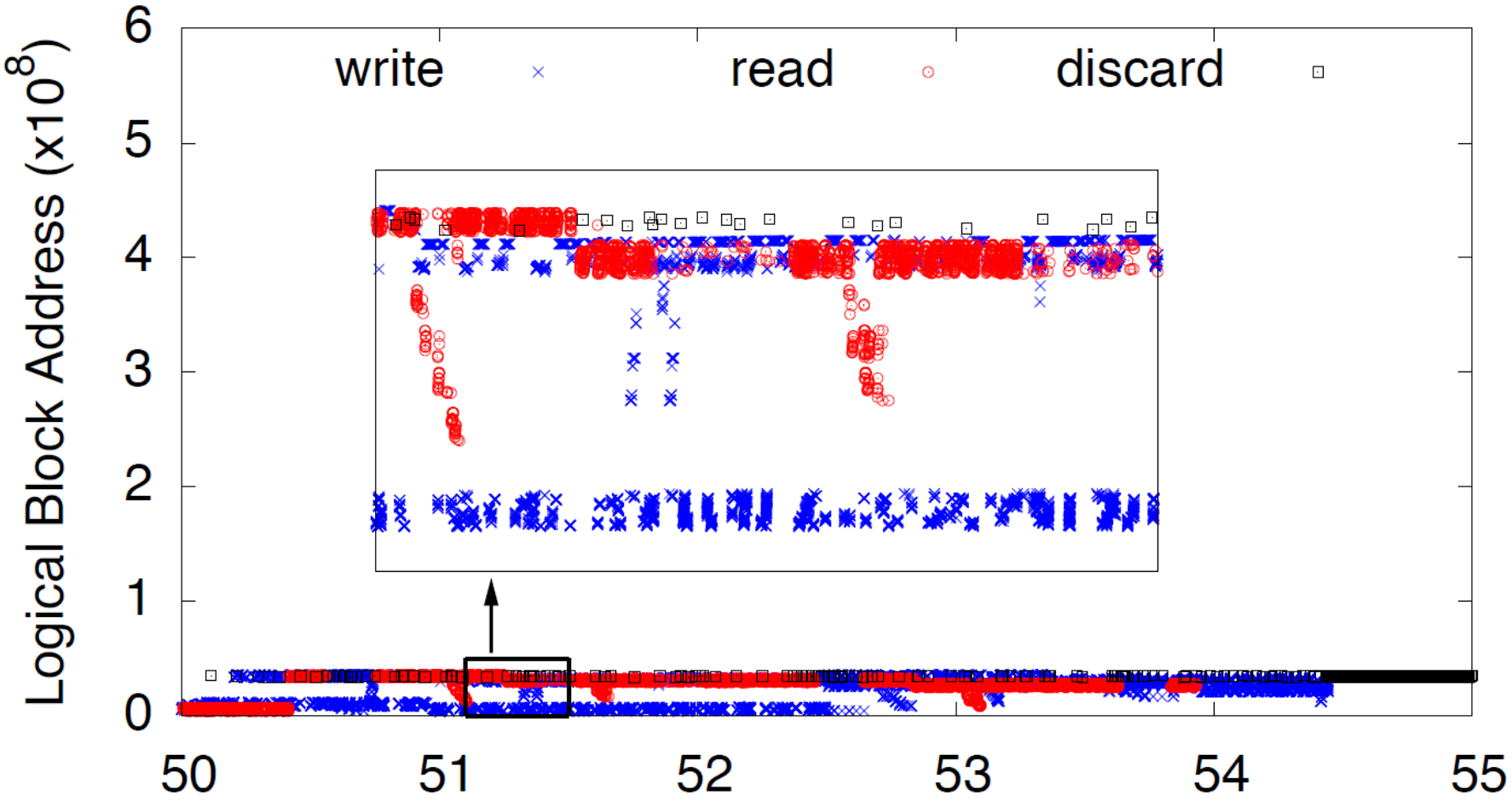


(b) PCIe SSD

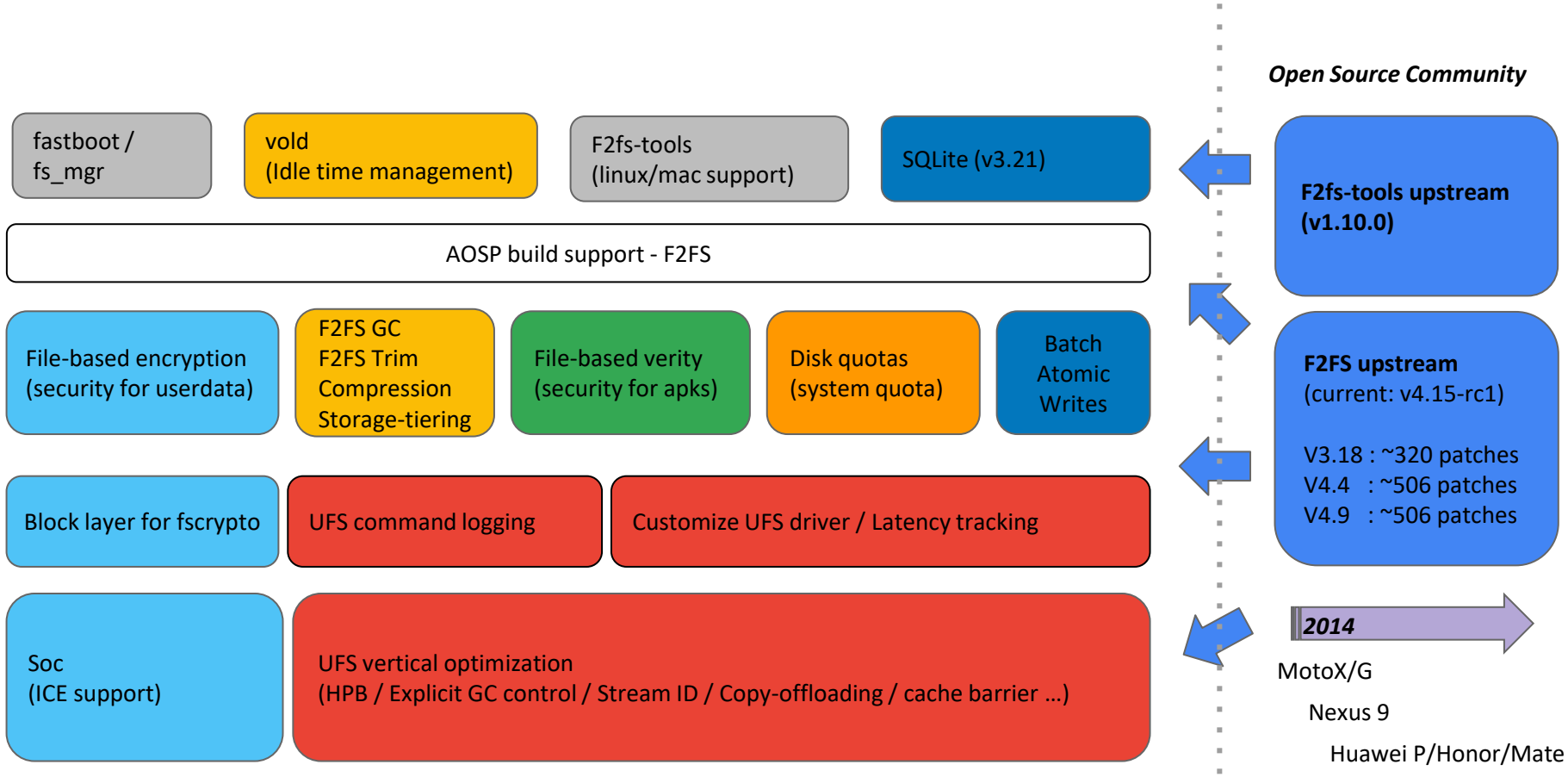
Storage Access Pattern: F2FS ("fileserver")



Storage Access Pattern: Ext4 ("fileserver")



Features for Android



F2FS on Android

SAMMOBILE NEWS SAMSUNG REVIEWS PREMIUM FIRMWARE

ONE UI 2.5 ONE UI 3.0 SECURITY UPDATE ANDROID 11 GALAXY NOTE 20 ULTRA

Last updated: August 12th, 2019 at 15:38 UTC+01:00

Galaxy Note 10 uses F2FS, not EXT4 file system: What's the difference?

This year, a couple of Samsung smartphones will make use of a file system other than EXT4, which has been traditionally used on the Android platform almost exclusively throughout its existence. The Galaxy Note 10 series, however, switches gears and adopts the F2FS file system together with UFS 3.0 storage. Today we'll explain some of the benefits to this, and what Samsung fans can expect in the future.