NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories

Andiry Xu, Steven Swanson

Non-volatile Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego

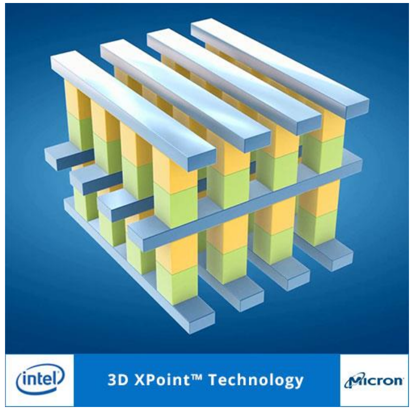
NOVA overview

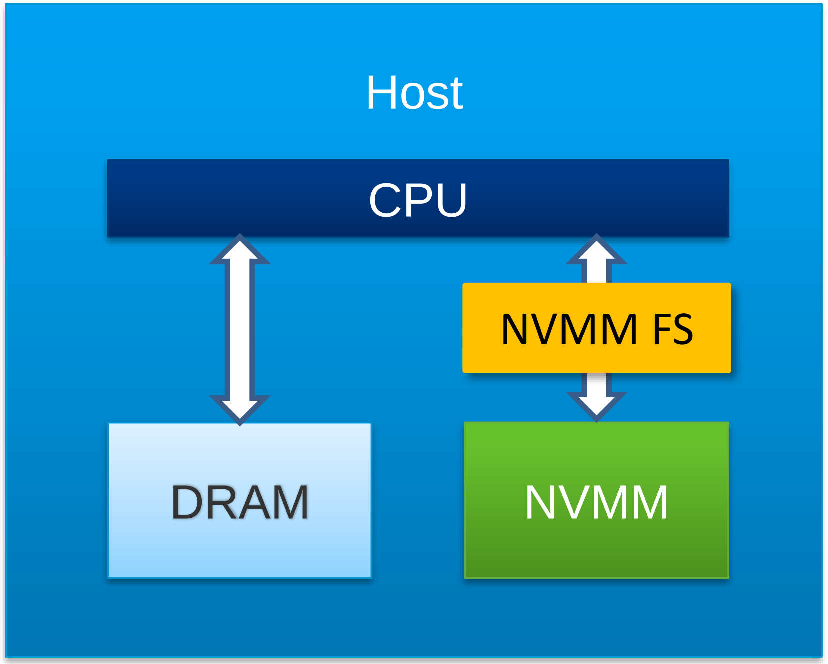
- NOVA extends LFS to leverage non-volatile memories
- NOVA proposes per-inode logging
- High performance + Strong atomicity
  - 3.1x to 13.5x to file systems that have equally strong consistency guarantees in write-intensive workloads
- POSIX compliant

https://github.com/NVSL/NOVA

Hybrid DRAM/NVMM system

- Non-volatile main memory (NVMM)
  - PCM, STT-RAM, ReRAM, 3D XPoint technology
- File system for NVMM





Disk-based file systems are inadequate for NVMM

- Ext4, xfs, Btrfs, F2FS, NILFS2
- Built for hard disks and SSDs
  - Software overhead is high
  - CPU may reorder writes to NVMM
  - NVMM has different atomicity guarantees

- Cannot exploit NVMM performance
- Performance optimization compromises consistency on system failure [1]

Atomicity	Ext4 wb	Ext4 order	Ext4 dataj	Btrfs	xfs
1-Sector overwrite	✓	✓	✓	✓	✓
1-Sector append	✗	✓	✓	✓	✓
1-Block overwrite	✗	✗	✓	✓	✗
1-Block append	✗	✓	✓	✓	✓
N-Block write/append	✗	✗	✗	✗	✗
N-Block prefix/append	✗	✓	✓	✓	✓

[1] Pillai *et al*, All File Systems Are Not Created Equal: On the Complexity of Crafting Crash-Consistent Applications, OSDI '14.

NVMM file systems are not strongly consistent

- BPFS, PMFS, Ext4-DAX, SCMFS, Aerie
- None of them provide strong metadata and data consistency

File system	Metadata atomicity	Data atomicity	Mmap Atomicity [1]
BPFS	Yes	Yes [2]	No
PMFS	Yes	No	No
Ext4-DAX	Yes	No	No
SCMFS	No	No	No
Aerie	Yes	No	No
NOVA	Yes	Yes	Yes

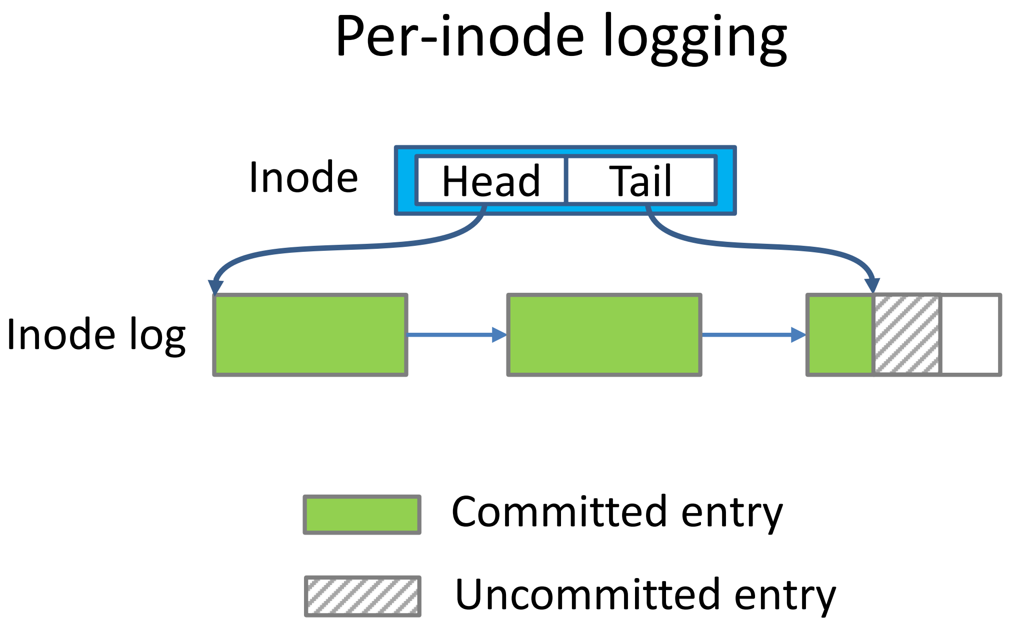
[1] Each msync() commits updates atomically.
[2] In BPFS, write times are not updated atomically with respect to the write itself.

Why LFS?

- Log-structuring provides cheaper atomicity than journaling and shadow paging
- NVMM supports fast, highly concurrent random accesses
  - Using multiple logs does not negatively impact performance
  - Log does not need to be contiguous
- Rethink and redesign log-structuring entirely

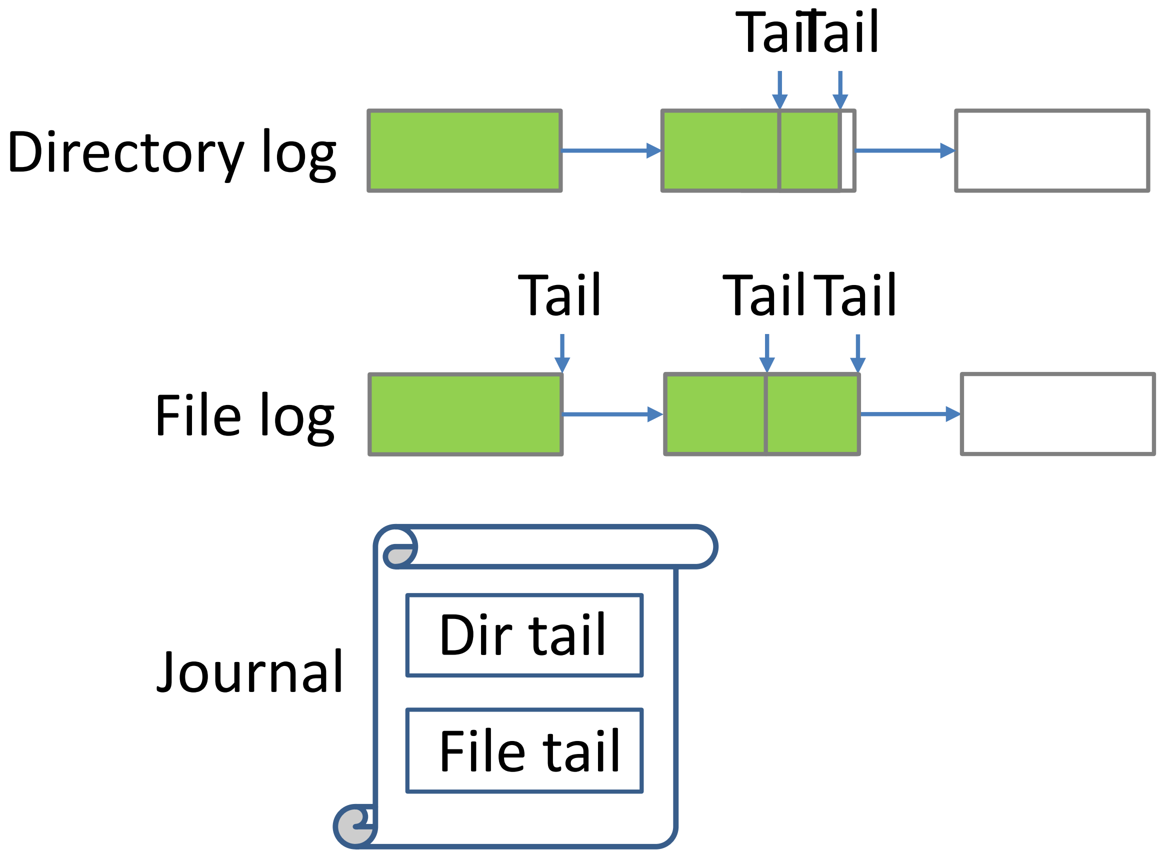
NOVA design goals

- Atomicity
  - Combine log-structuring, journaling and copy-on-write
- High performance
  - Split data structure between DRAM and NVMM
  - Highly scalable
- Efficient garbage collection
  - Fine-grained log cleaning with log as a linked list
  - Log only contains metadata
- Fast recovery
  - Lazy rebuild
  - Parallel scan



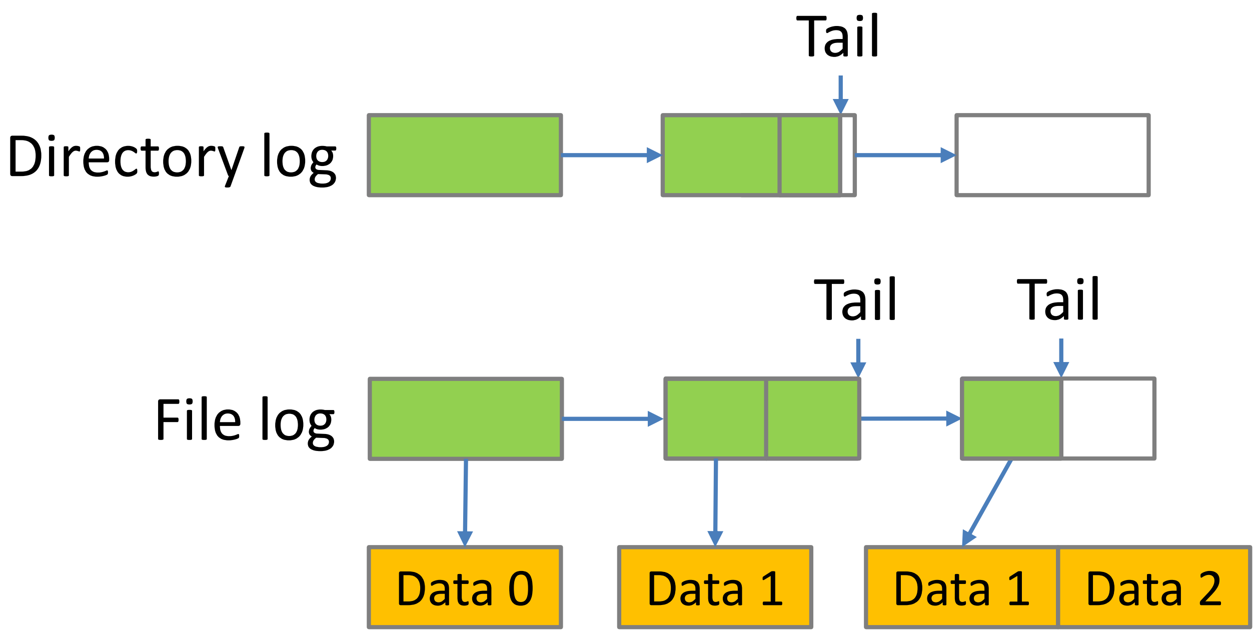
Atomicity

- Log-structuring for single log update
  - Write, msync, chmod, etc
  - Strictly commit log entry to NVMM before updating log tail
- Lightweight journaling for update across logs
  - Unlink, rename, etc
  - Journal log tails instead of metadata or data
- Copy-on-write for file data
  - Log only contains metadata
  - Log is short



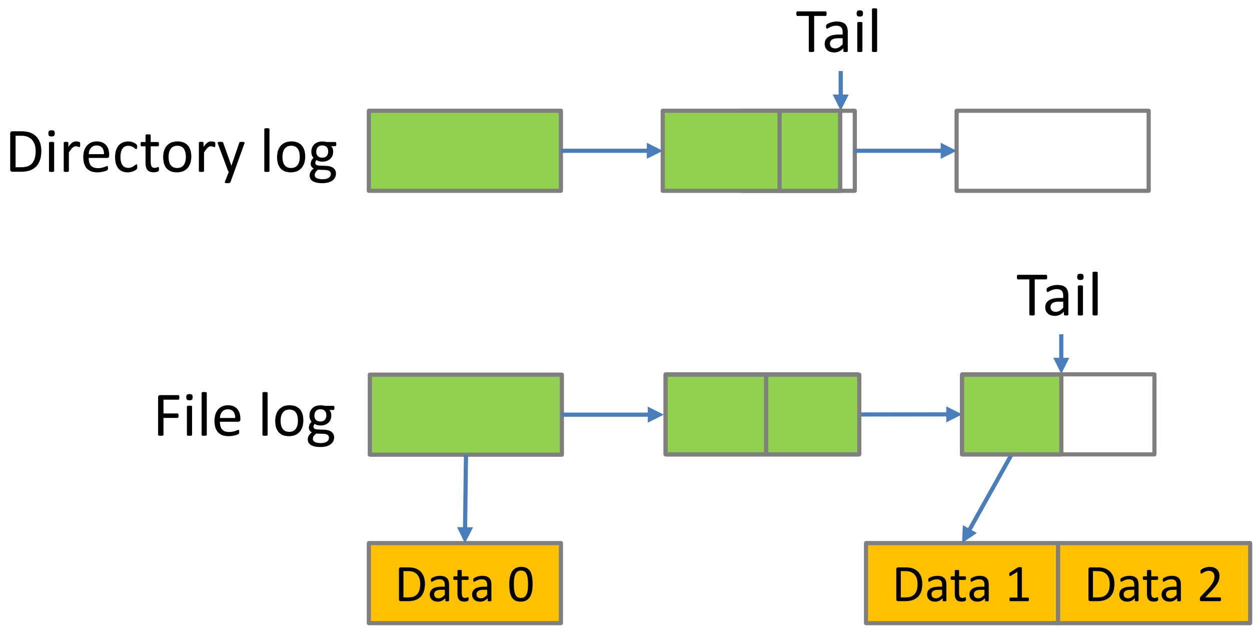
Atomicity

- Log-structuring for single log update
  - Write, msync, chmod, etc
  - Strictly commit log entry to NVMM before updating log tail
- Lightweight journaling for update across logs
  - Unlink, rename, etc
  - Journal log tails instead of metadata or data
- Copy-on-write for file data
  - Log only contains metadata
  - Log is short



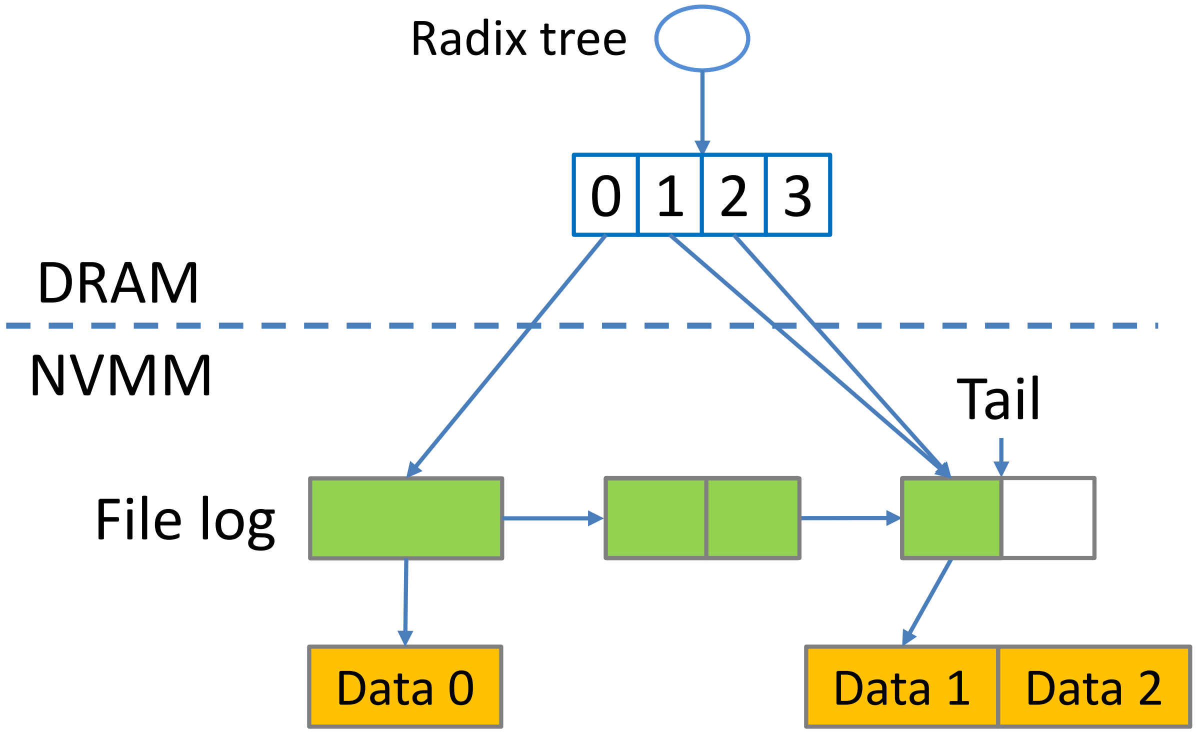
Performance

- Per-inode logging allows for high concurrency
- Split data structure between DRAM and NVMM
  - Persistent log is simple and efficient
  - Volatile tree structure has no consistency overhead



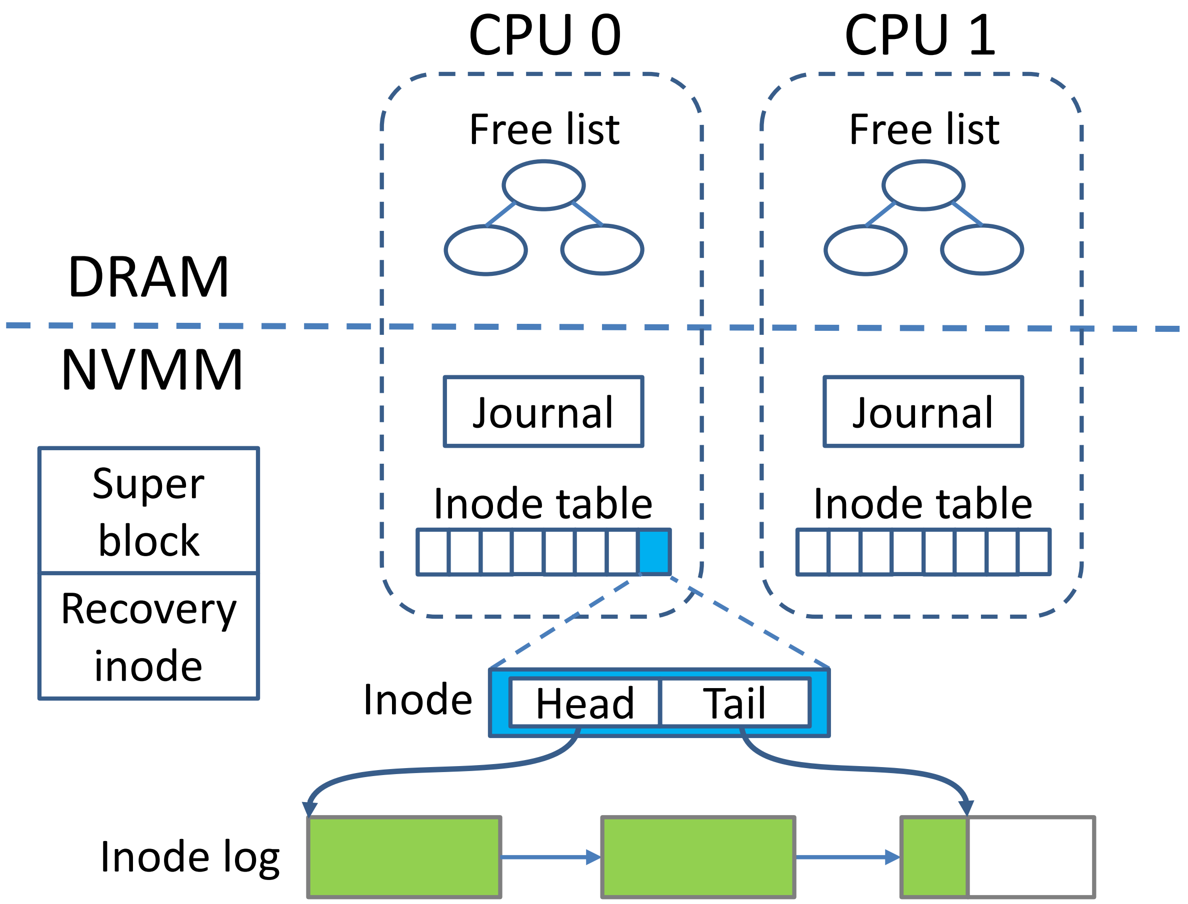
Performance

- Per-inode logging allows for high concurrency
- Split data structure between DRAM and NVMM
  - Persistent log is simple and efficient
  - Volatile tree structure has no consistency overhead



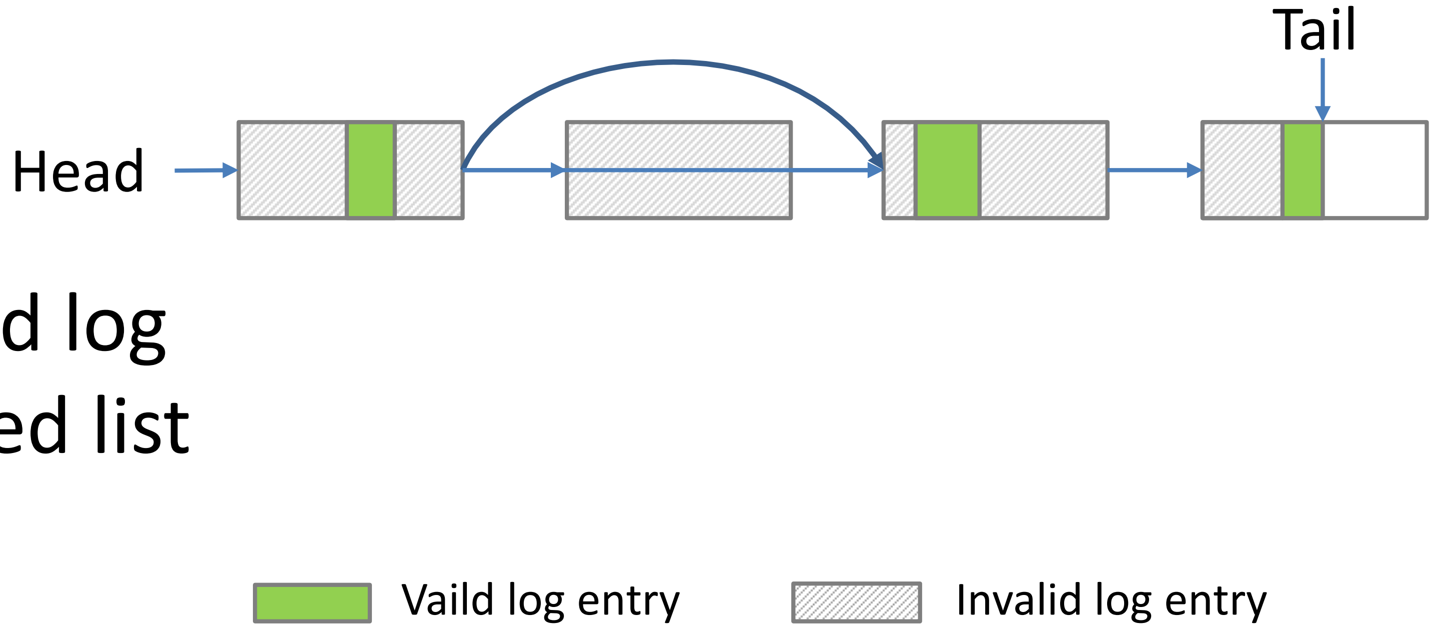
NOVA layout

- Put allocator in DRAM
- High scalability
  - Per-CPU NVMM free list, journal and inode table
  - Concurrent transactions and allocation/deallocation



Fast garbage collection

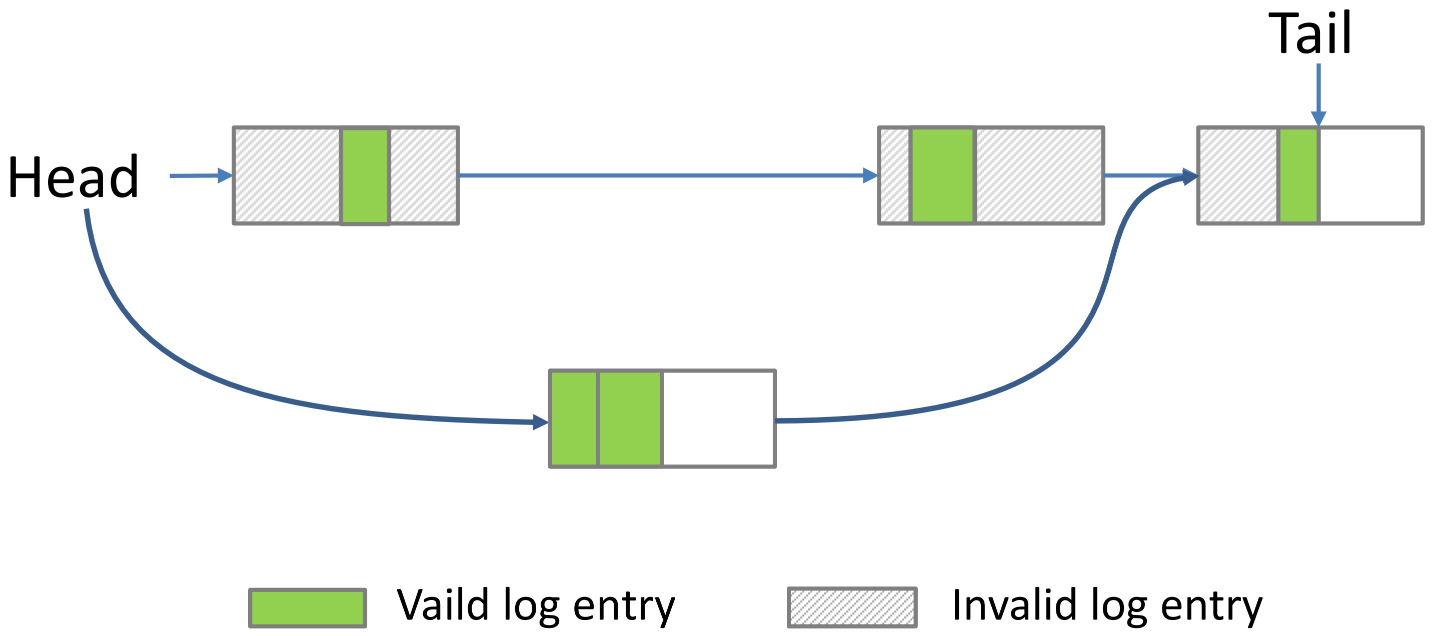
- Log is a linked list
- Log only contains metadata



- Fast GC deletes dead log pages from the linked list
- No copying

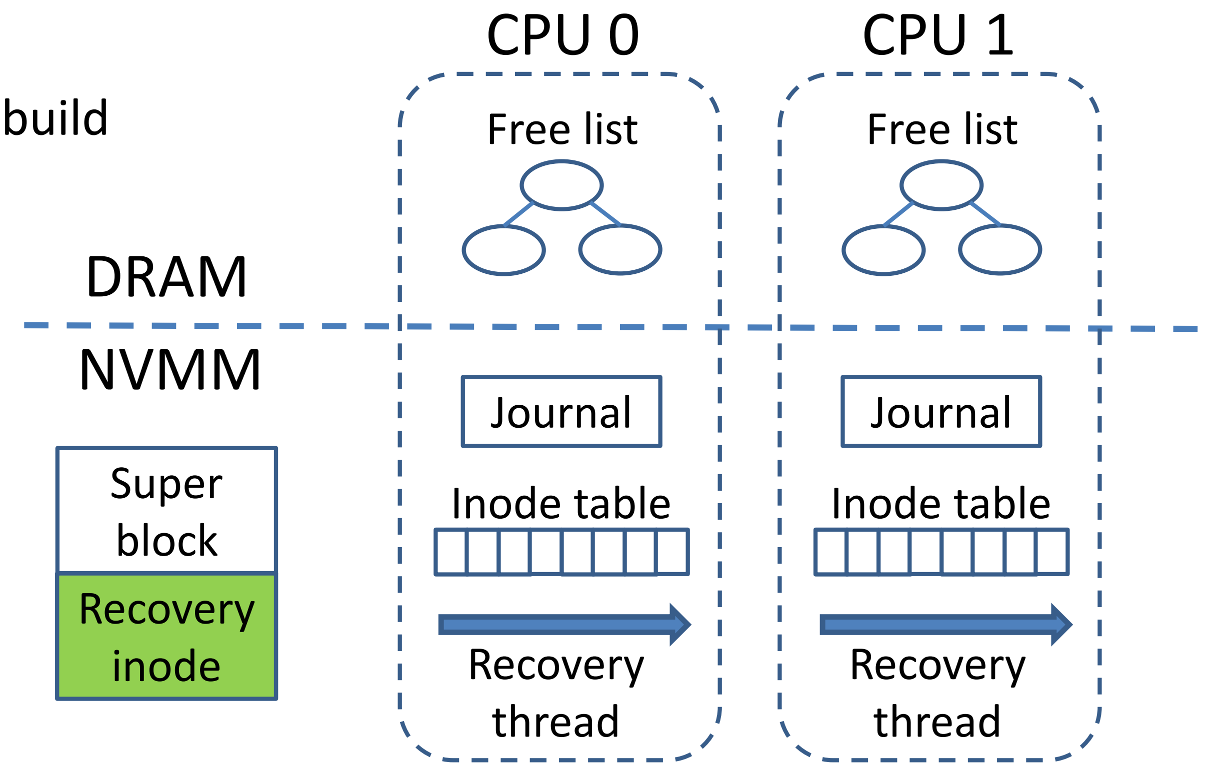
Thorough garbage collection

- Starts if valid log entries < 50% log length
- Format a new log and atomically replace the old one
- Only copy metadata

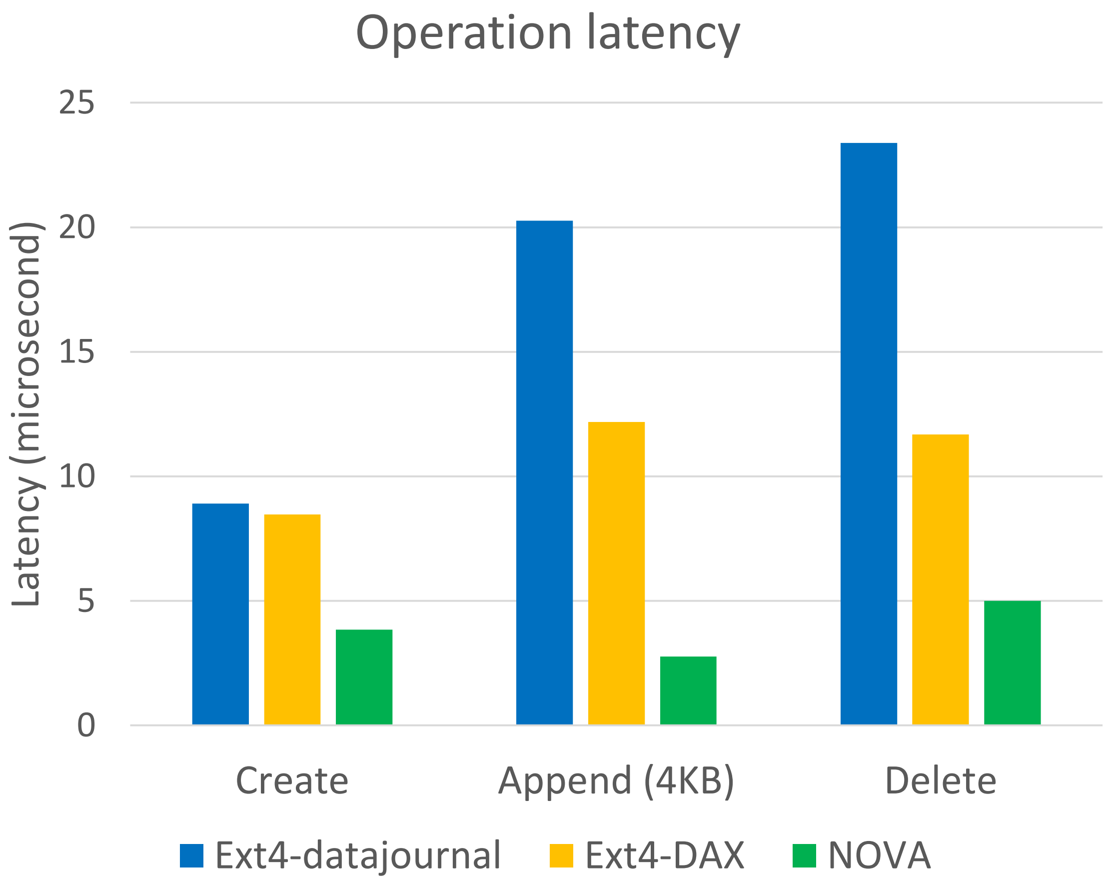


Recovery

- Rebuild DRAM structure
  - Allocator
  - Lazy rebuild: postpones inode radix tree rebuild
    - Accelerates recovery
    - Reduces DRAM consumption
- Normal shutdown recovery:
  - Store allocator in recovery inode
  - No log scanning
- Failure recovery:
  - Log is short
  - Parallel scan
  - Failure recovery bandwidth: > 400 GB/s

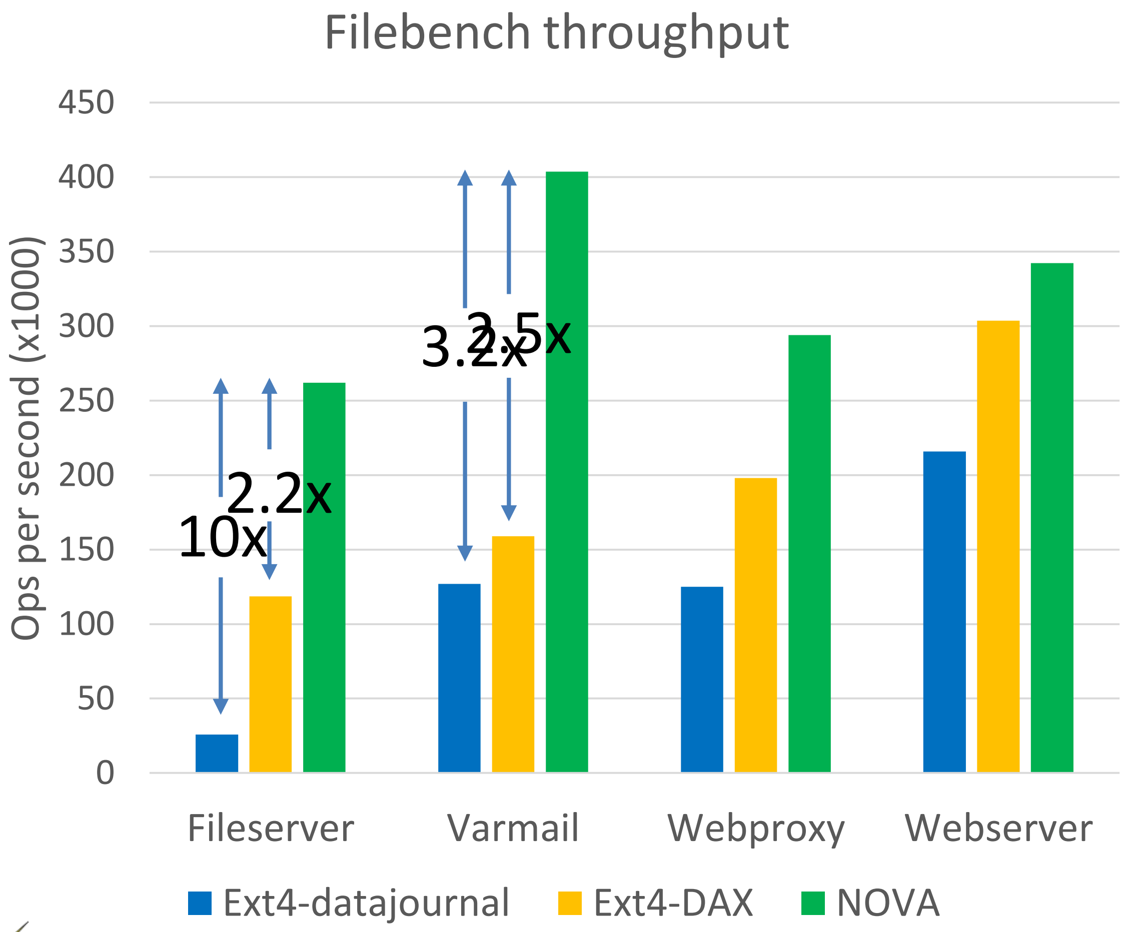


Evaluation: Latency



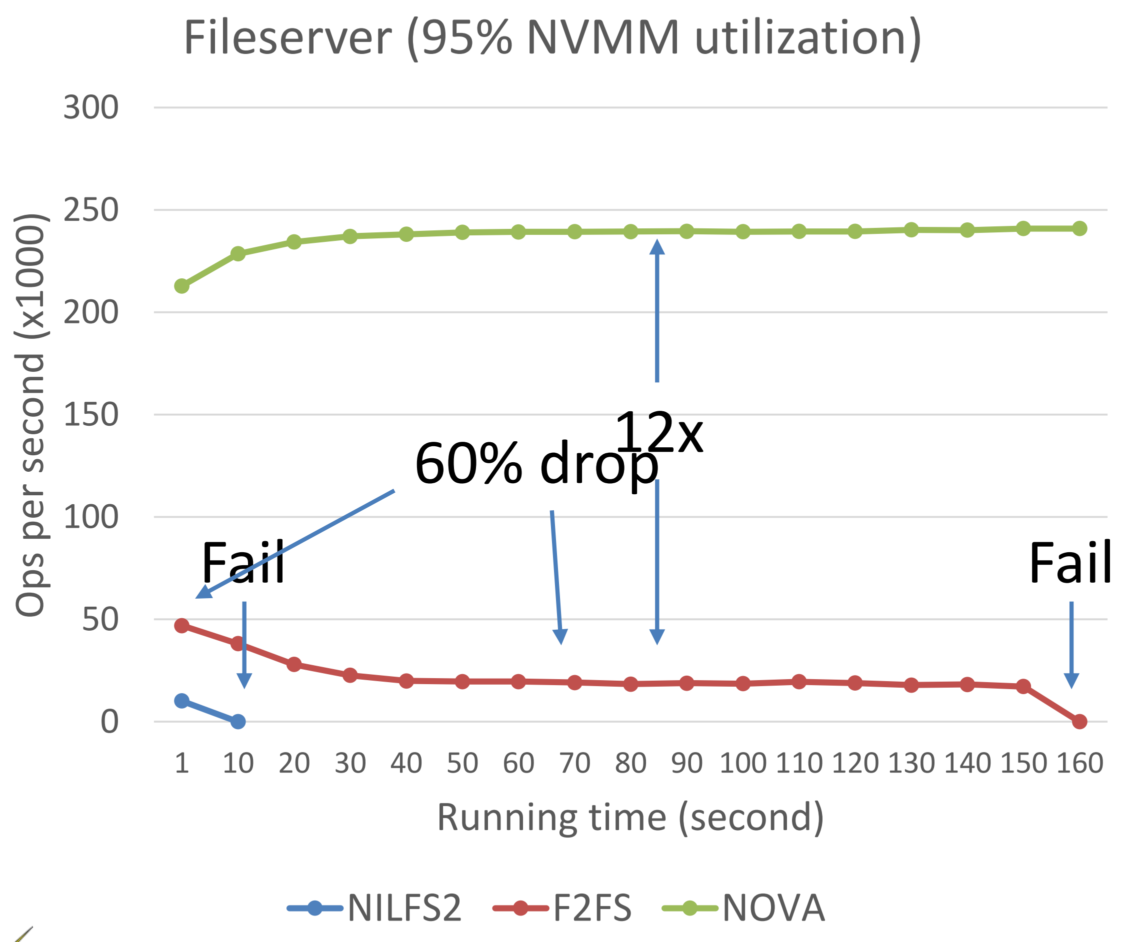
- Intel PM Emulation Platform
  - Emulates different NVM characteristics
  - Emulates clwb/PCOMMIT latency
- NOVA provides low latency atomicity

Filebench throughput



- NOVA achieves high performance with strong data consistency

Garbage collection efficiency



- NOVA's performance stays stable with increasing running time
- Fast GC reclaims the majority of stale pages in the long-term running

Conclusion

- Existing file systems do not meet the requirements of applications on NVMM file systems
- NOVA’s multi-log design achieves high concurrency, efficient garbage collection and fast recovery
- NOVA outperforms existing file systems while providing stronger consistency and atomicity guarantees

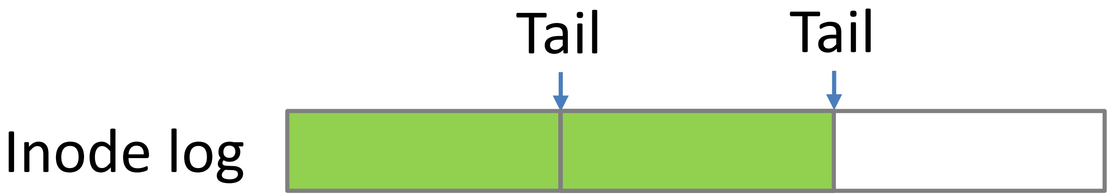
Thank you!

https://github.com/NVSL/NOVA

Backup slides

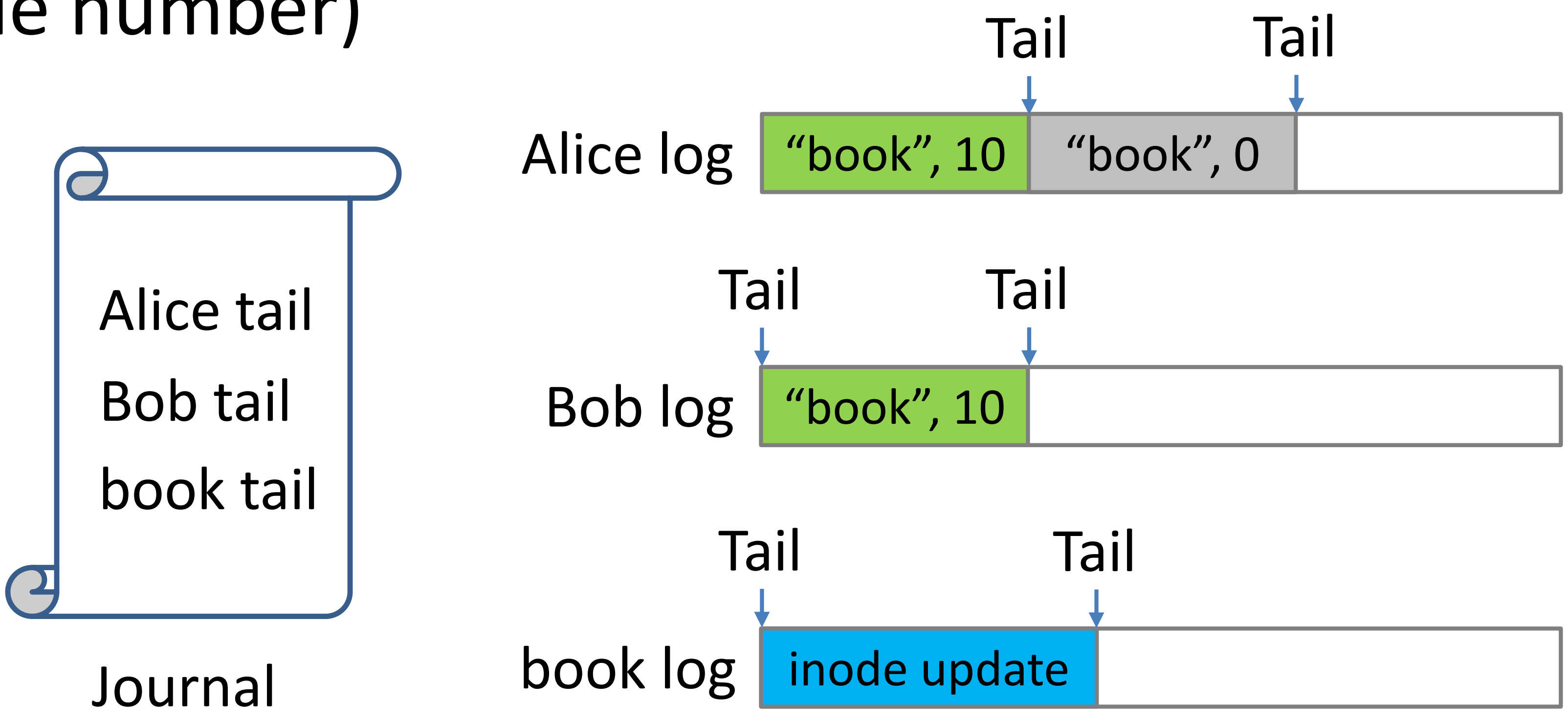
Atomicity and enforce write ordering

```
// Strictly commit log entry to NVMM before updating tail
new_tail = append_to_log(inode->tail, entry);
clwb(inode->tail, entry->length);        // writes back the cachelines
sfence();
PCOMMIT();          // Commits to NVMM
sfence();
inode->tail = new_tail;
```



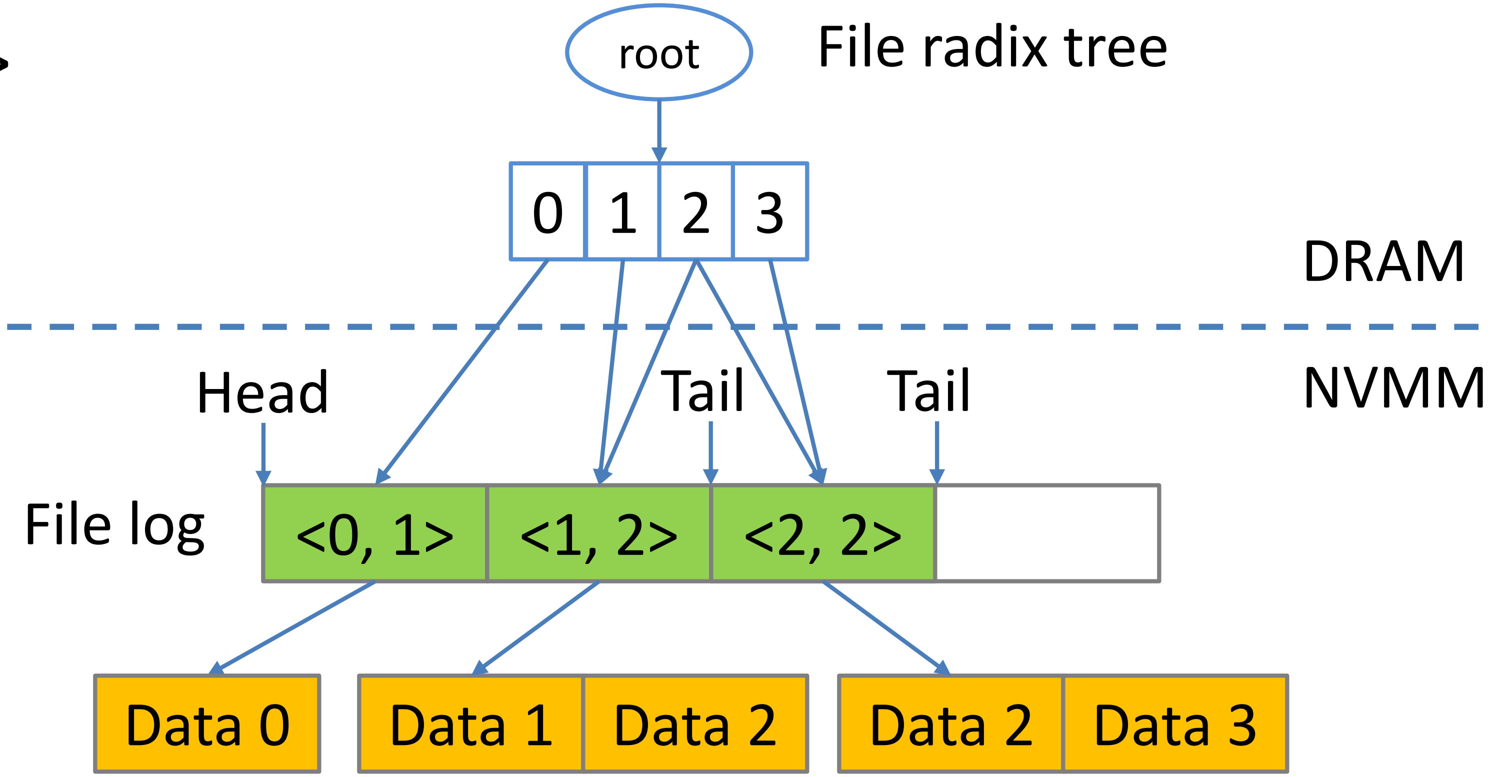
Directory operations

- mv Alice/book Bob/
- (name, inode number)



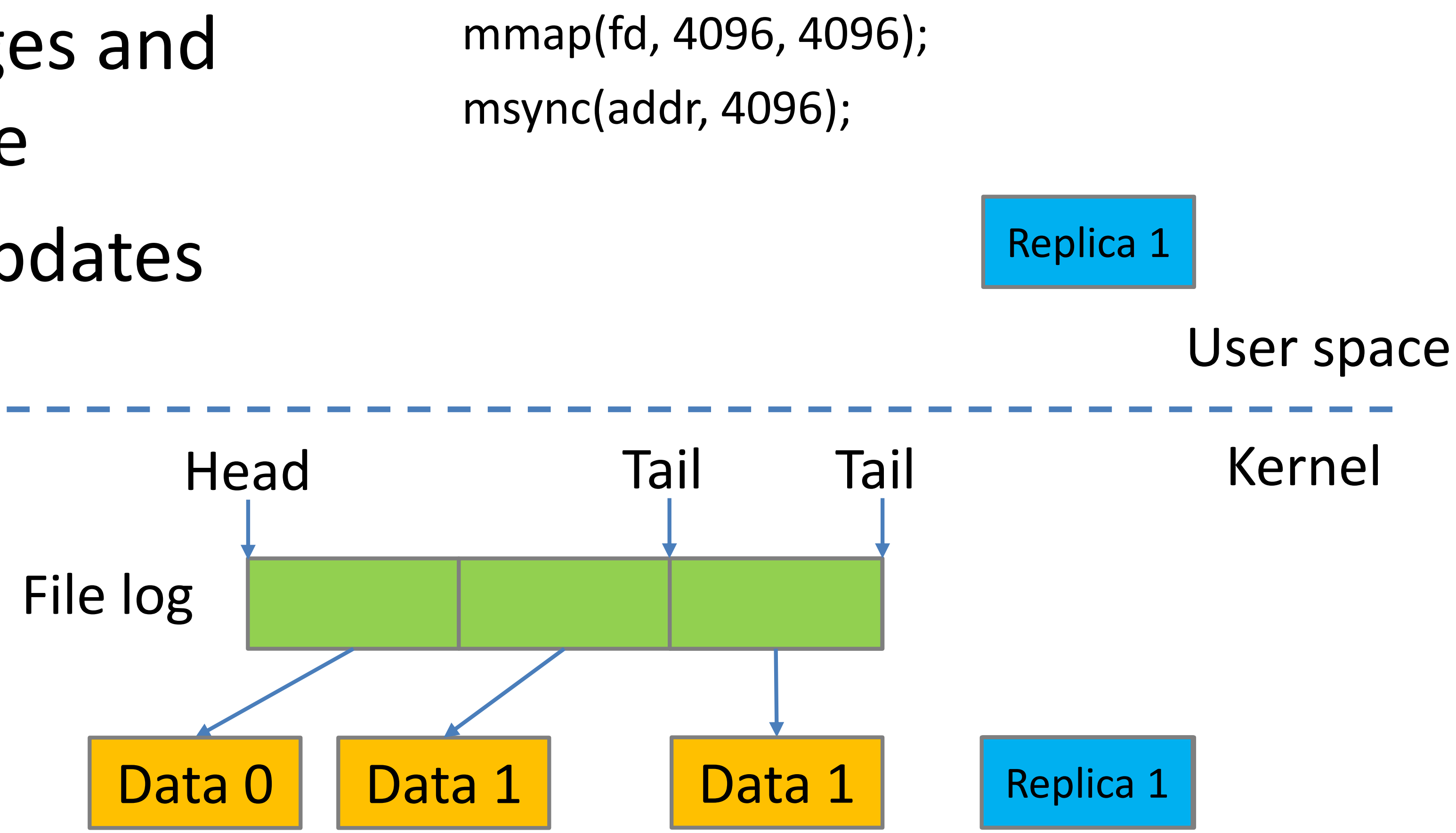
Atomic file operations

- Copy-on-write for file data
- <pgoff, num pages>
- Write(8192, 8192)



Atomic mmap

- Allocate replica pages and mmap to user space
- msync() commits updates atomically



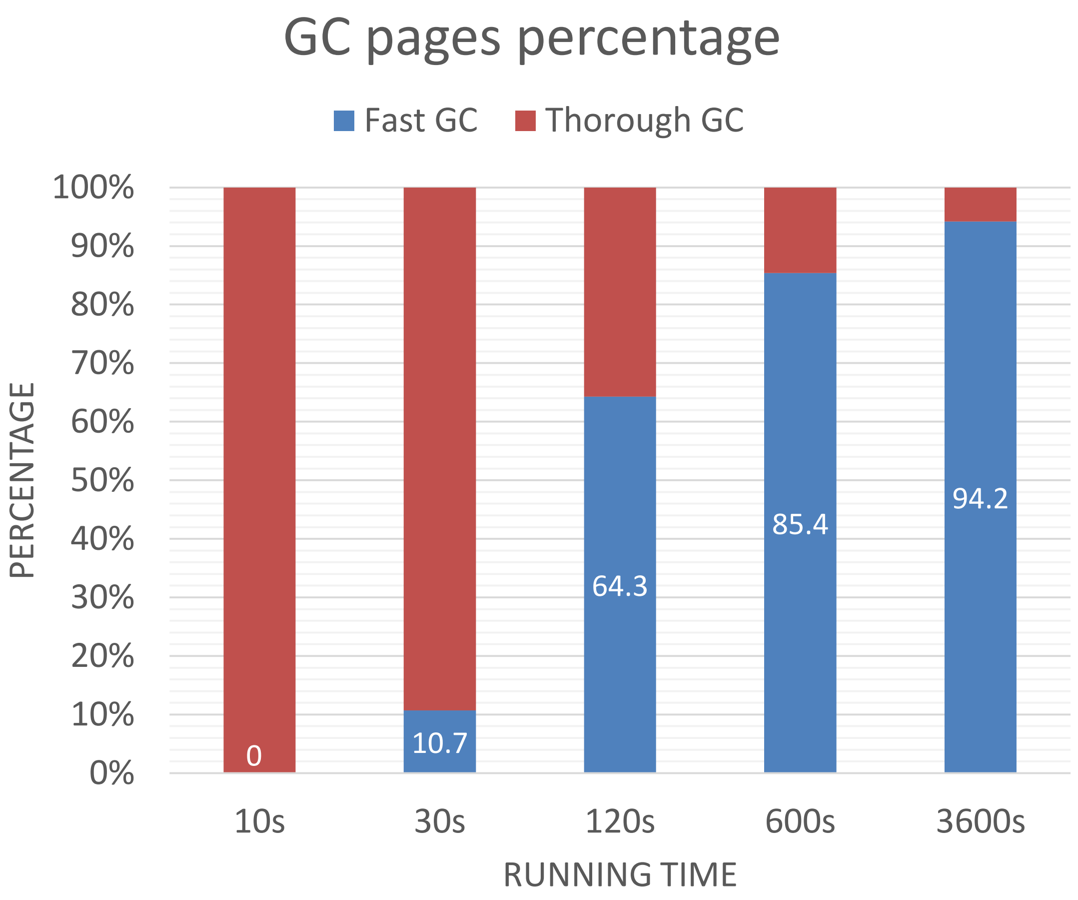
Evaluation

- Intel PM Emulation Platform
- 32GB of DRAM, 64GB of NVMM

NVMM	Read latency	Write bandwidth	clwb latency	PCOMMIT latency
STT-RAM	100 ns	Full DRAM	40 ns	200 ns
PCM	300 ns	1/8 DRAM	40 ns	500 ns

- Compare to Btrfs, NILFS2, F2FS, Ext4, Ext4-data, Ext4-DAX, PMFS
- Linux kernel 4.0 x86-64

Garbage collection efficiency



- Fast GC reclaims 94% pages in one-hour test