Libnvmmio: Reconstructing SW IO Path with Failure-Atomic Memory-Mapped Interface

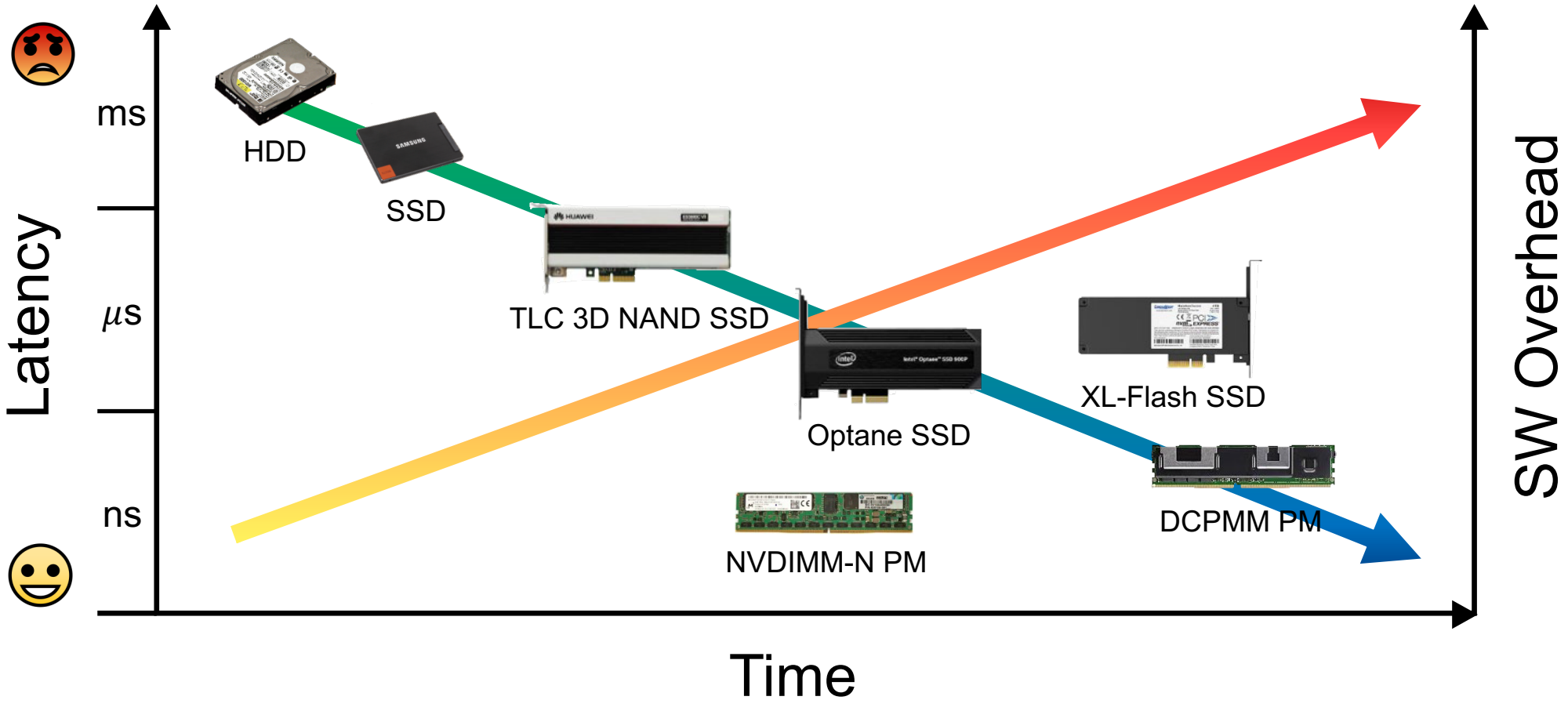
Jungsik Choi[1], Jaewan Hong[2], Youngjin Kwon[2], Hwansoo Han[1]

[1] SUNG KYUN KWAN UNIVERSITY

[2] KAIST

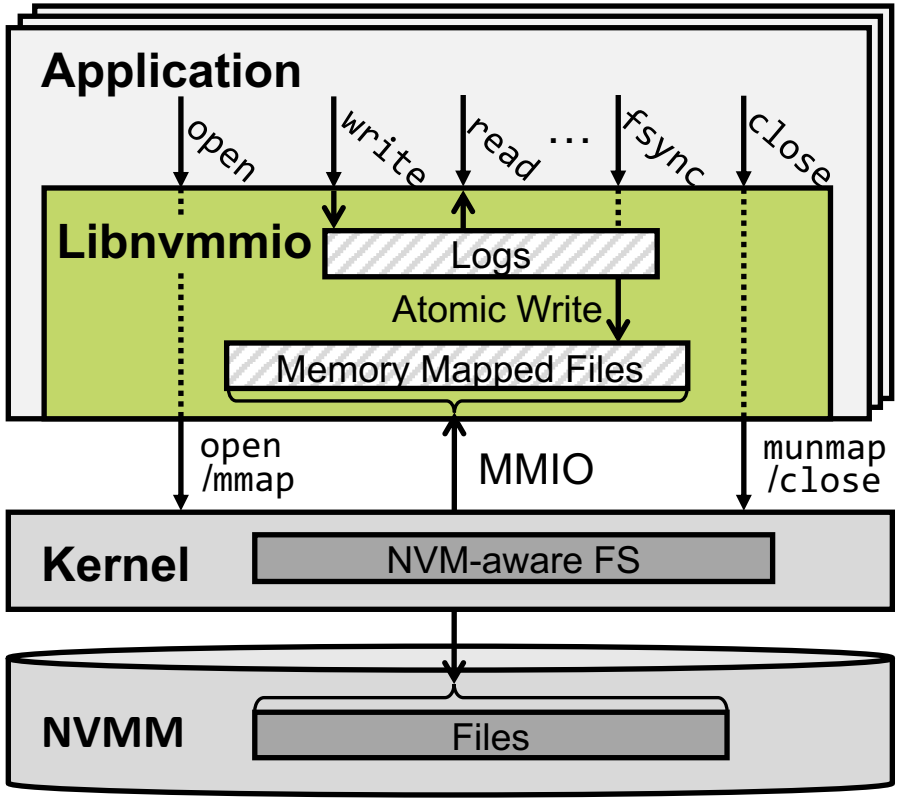
USENIX ATC '20

SW Overhead Greater than Storage Latency



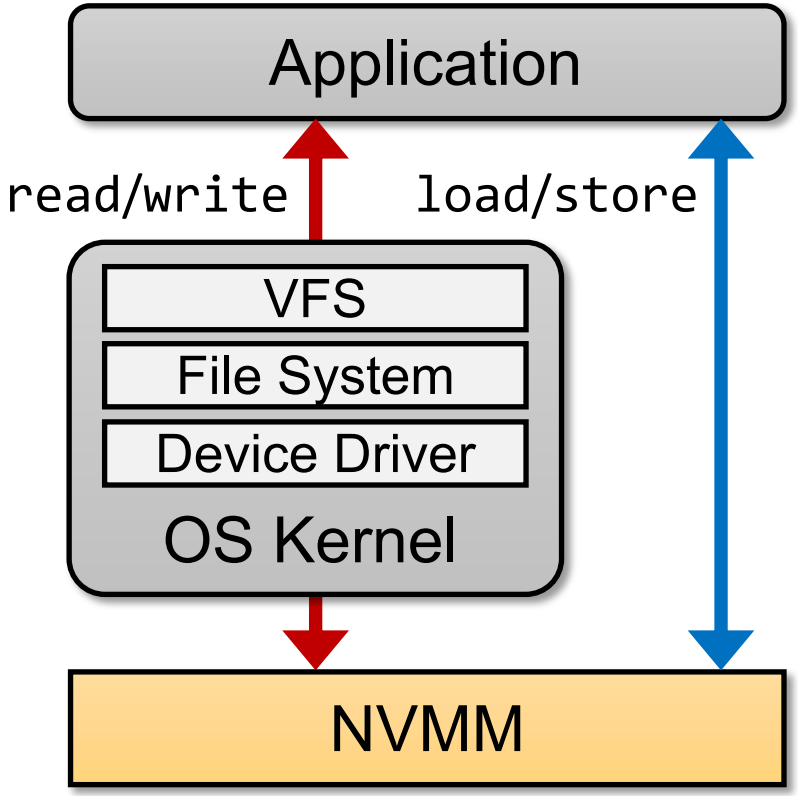
Reconstruct SW IO Path with Libnvmmio

- Libnvmmio
  - Library
  - Run on any POSIX FS (DAX-mmap)
  - Transparent MMIO with logging
  - Make common IO path efficient
    - Handle data ops at user-level
    - Route metadata ops to kernel FS
  - Low-latency & scalable IO
  - Data-atomicity



User-Level IO is Suitable in NVMM system

- Kernel's IO stacks introduce SW overhead
- User-level IO with `mmap`
  - Access files directly with `load/store`
  - Reduce user/kernel mode switches
  - Avoid complex IO stacks
  - No indexing, no permission checks
- MMIO is the fastest way to access files

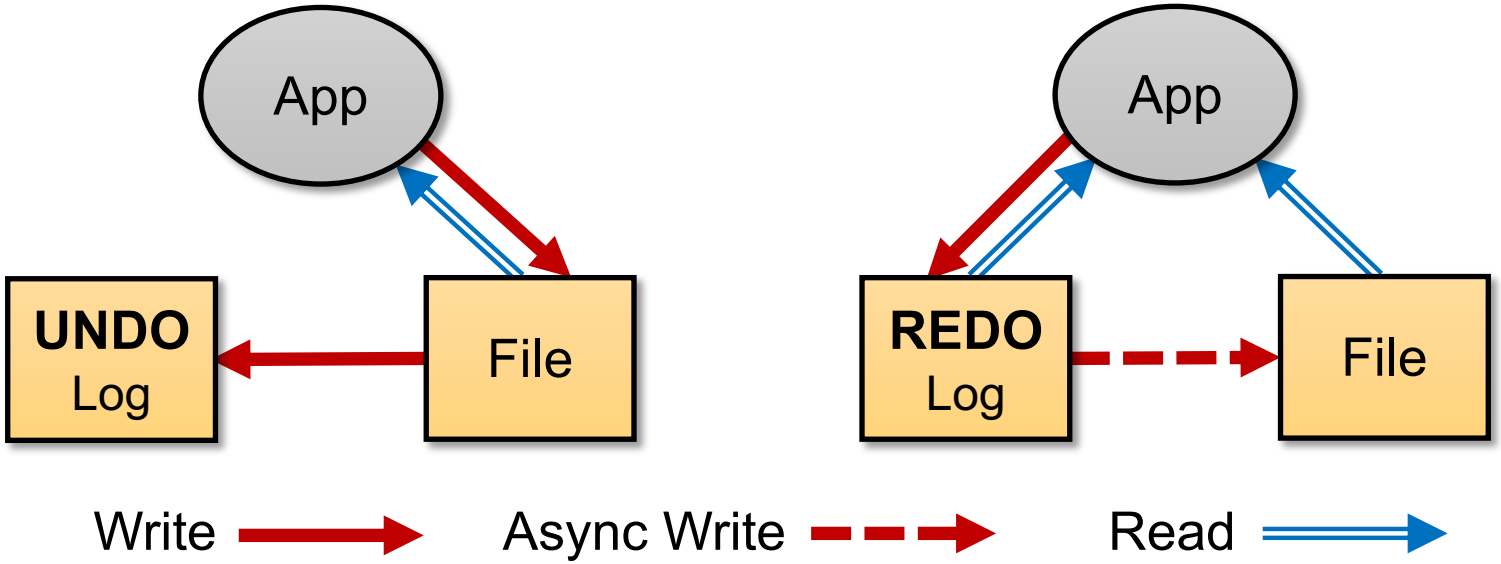


Logging is more Efficeint than CoW

- CoW (or shadow paging)
  - High write amplification
  - Hugepages make CoW more expensive
  - Frequent TLB-shutdown
- Logging (or journaling)
  - Writing data twice: logs and files
  - Differential logging
  - Checkpointing can be postponed

Redo *vs.* Undo

- Most logging systems use only one policy (redo or undo)
- They have different pros & cons depending on access type
  - REDO is better for writing, UNDO is better for reading

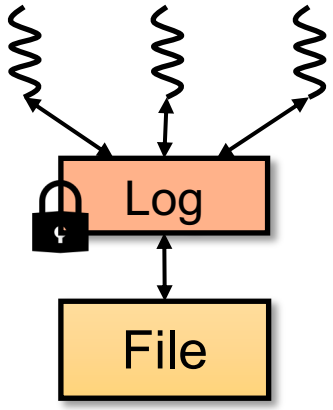


Hybrid Logging

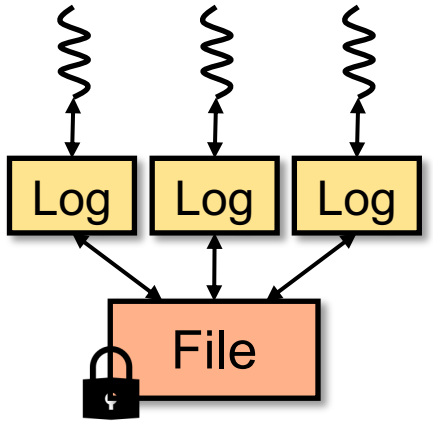
- Uses **adaptive policy** depending on the access type of a file
  - Read-intensive file → **Undo** logging
  - Write-intensive file → **Redo** logging
- Maintains per-file read/write counters
- Determines logging policy on each fsync
- Achieves the best case performance of two logging policies
  - Reduce SW overhead and improve logging efficiency

Centralized Logging with Fine-Grained Locks

- Decentralized logging was designed for transactions
  - *e.g.*, per-thread logging, per-transaction logging
- Centralized logging is appropriate for file IO, but not scalable
  - Requires fine-grained locks for scalable file IO

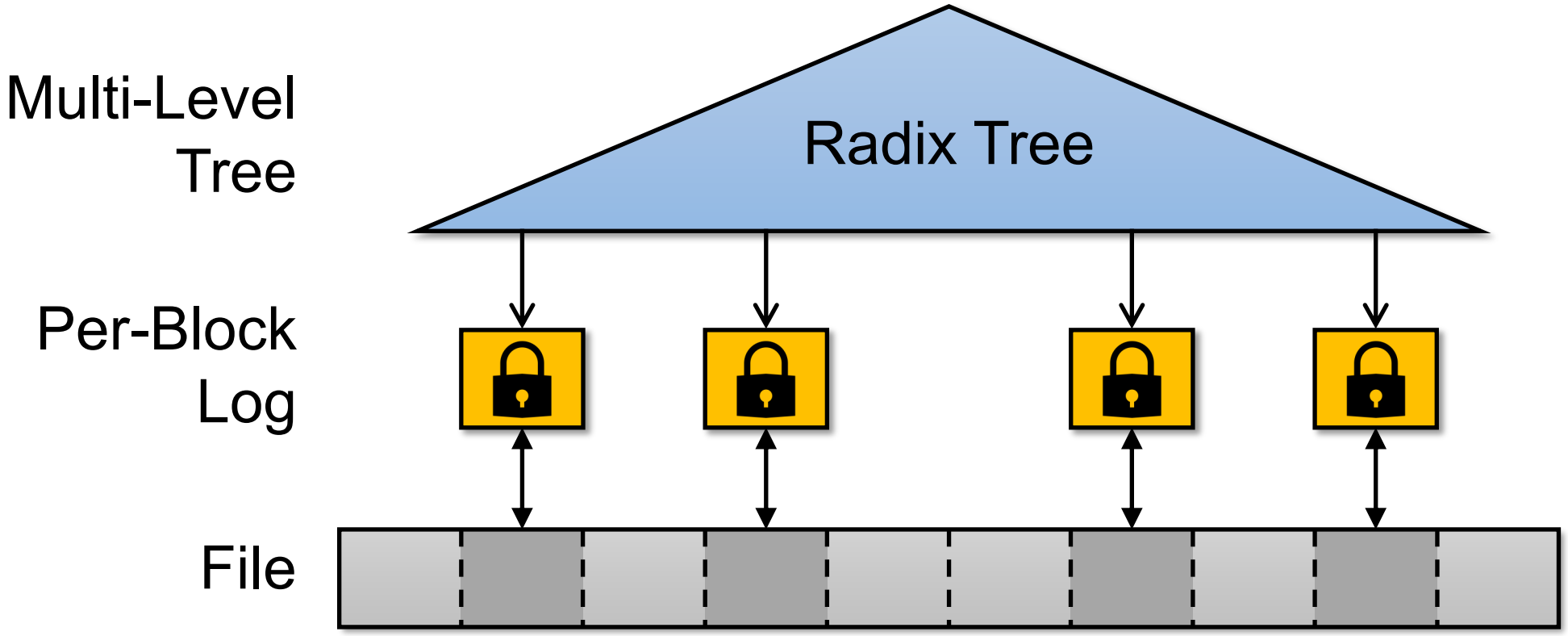


Centralized Logging

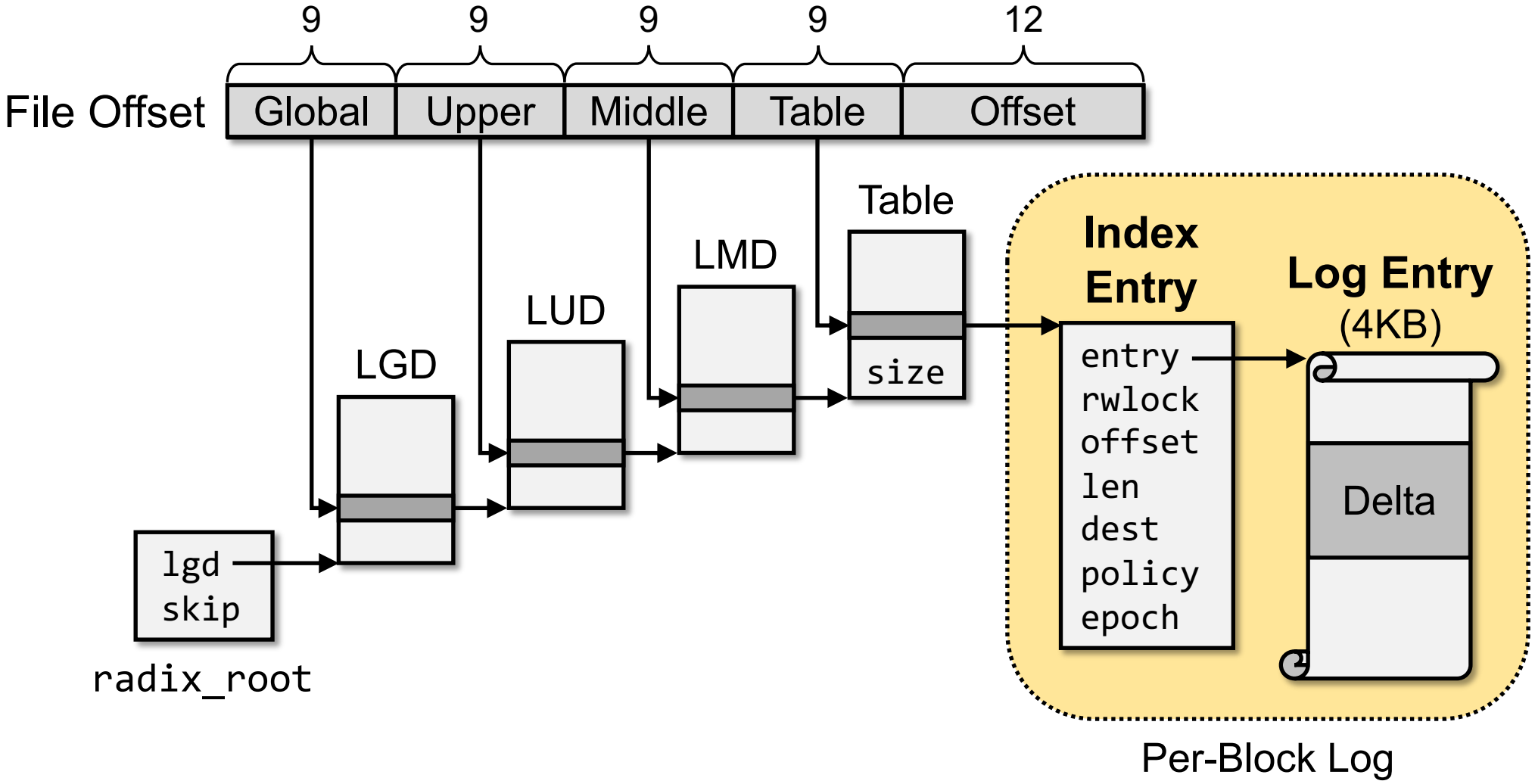


Decentralized Logging

Per-Block Logging

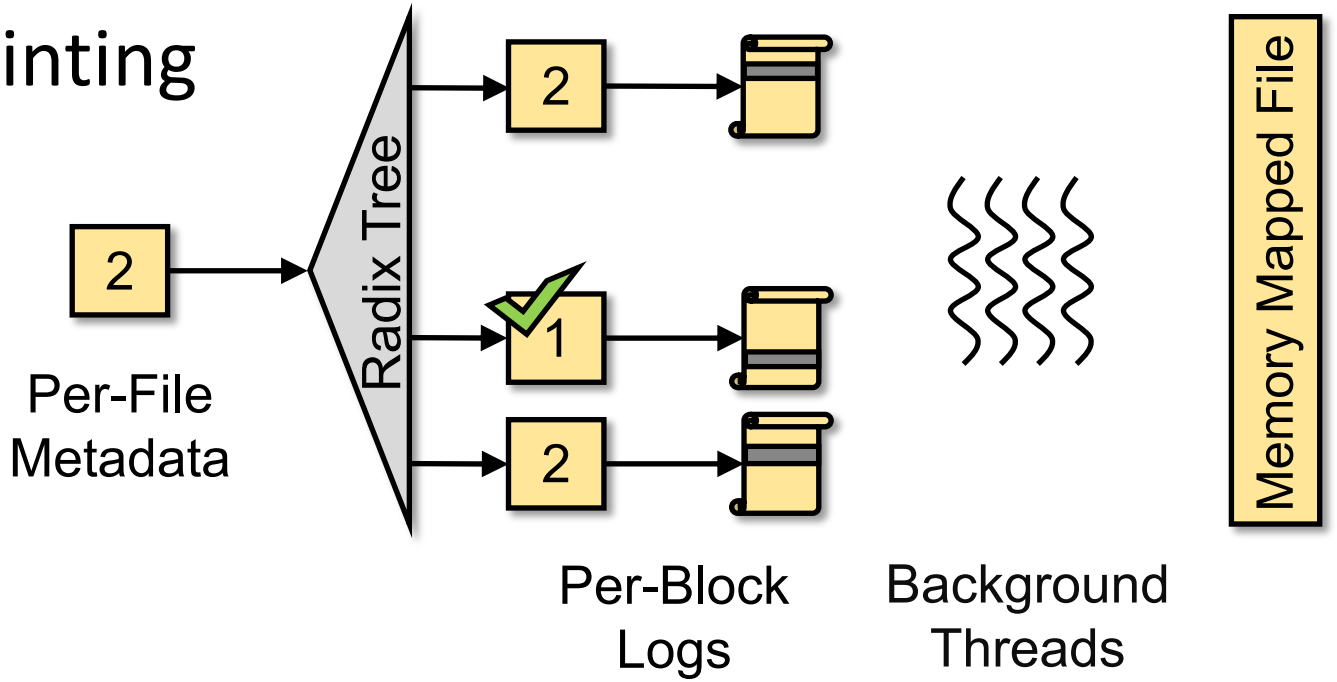


Lock-Free Radix Tree



Commit & Checkpoint based on Epoch

- Per-block logs are atomically committed on fsync
- Libnvmmio commits by increasing the global epoch value
  - Committed logs have an epoch smaller than the global epoch
- Background ckeckpointing



Design Summary

Libnvmmio provides low-latency and scalable IO
while guaranteeing data-atomicity

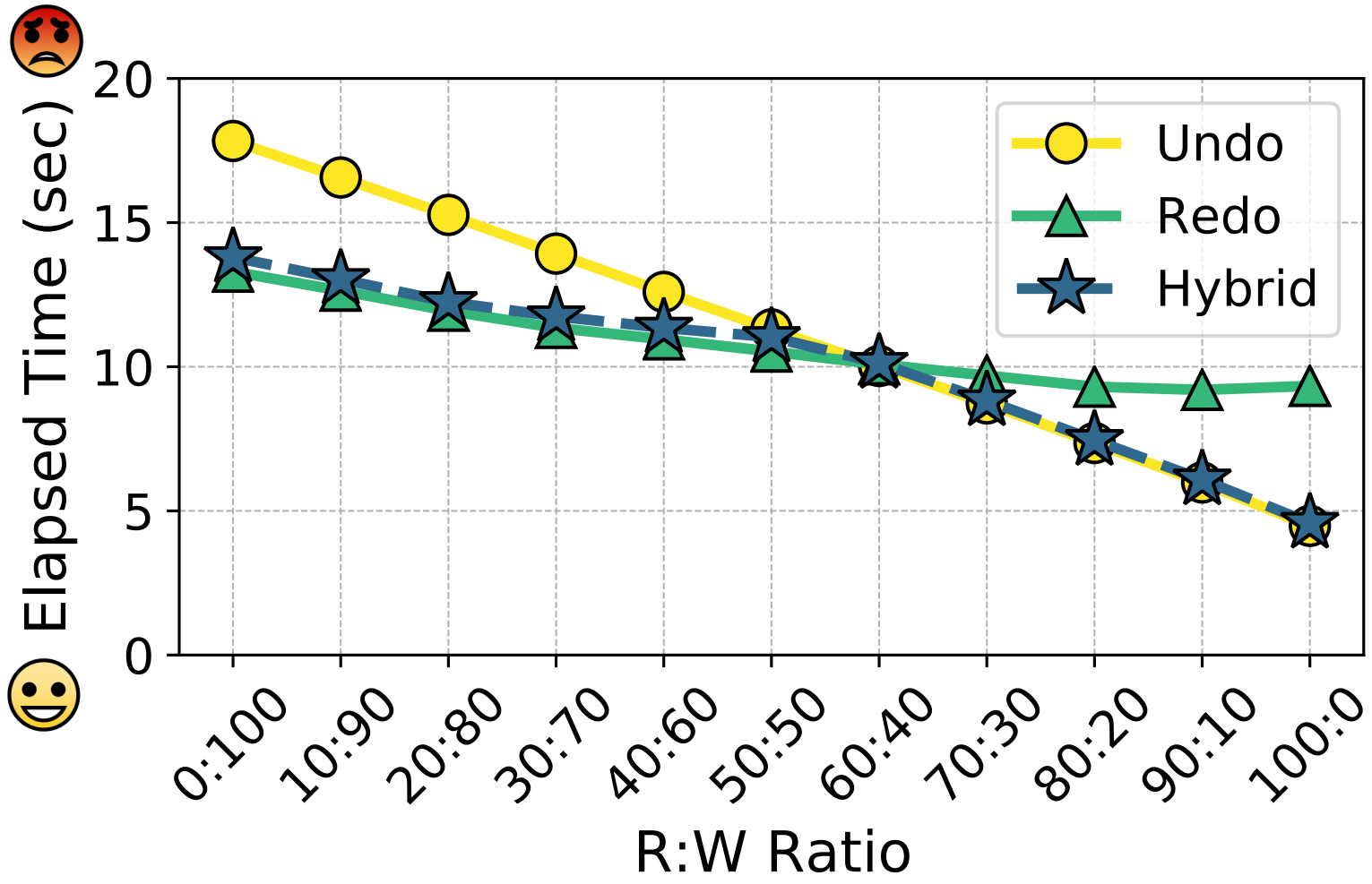
- Low-latency IO
  - User-level IO with mmap
  - Differential logging
  - Hybrid logging
  - Various log sizes
  - Epoch-based committing
  - Background checkpointing
- Scalable IO
  - Per-block logging
  - Lock-free index data structure

Experimental Setup

- Experimental Machines
  - 32GB **NVDIMM-N**, 20 cores and 32GB DRAM
  - 256GB **Optane DC**, 16 cores and 128GB DRAM (in our paper)
- Comparison systems

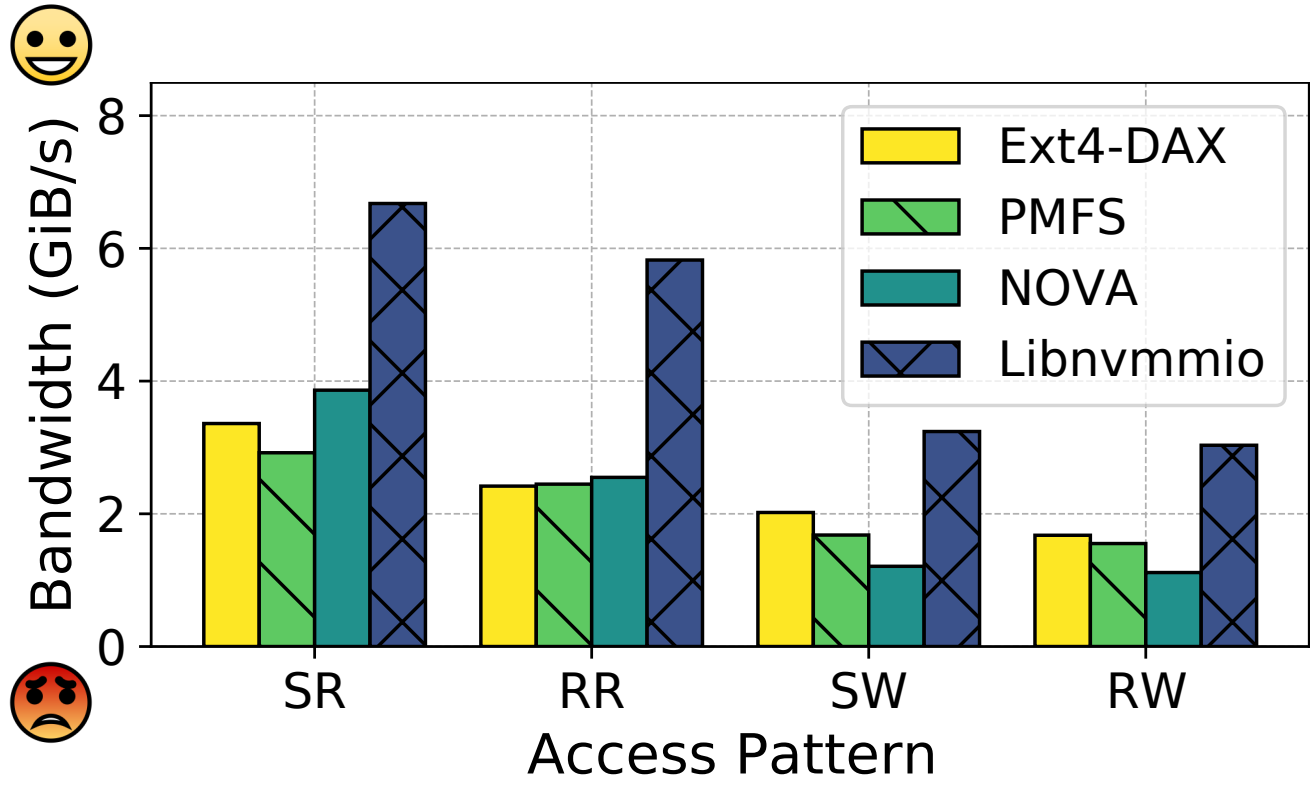
Filesystem	File IO	Data-Atomicity	Kernel
Ext4-DAX	Kernel	X	5.1
PMFS	Kernel	X	4.13
NOVA	Kernel	O	5.1
SplitFS	User	O	4.13
Libnvmmio*	User	O	5.1

Hybrid Logging

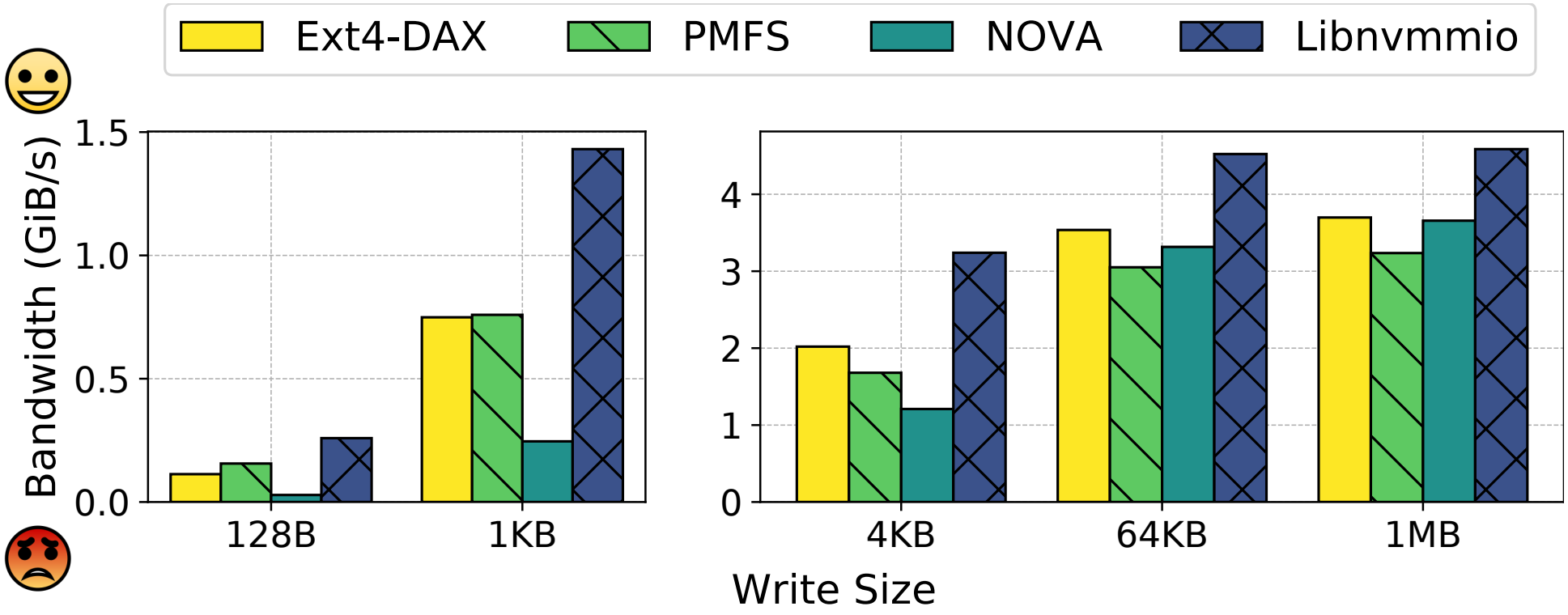


FIO: Different Access Patterns

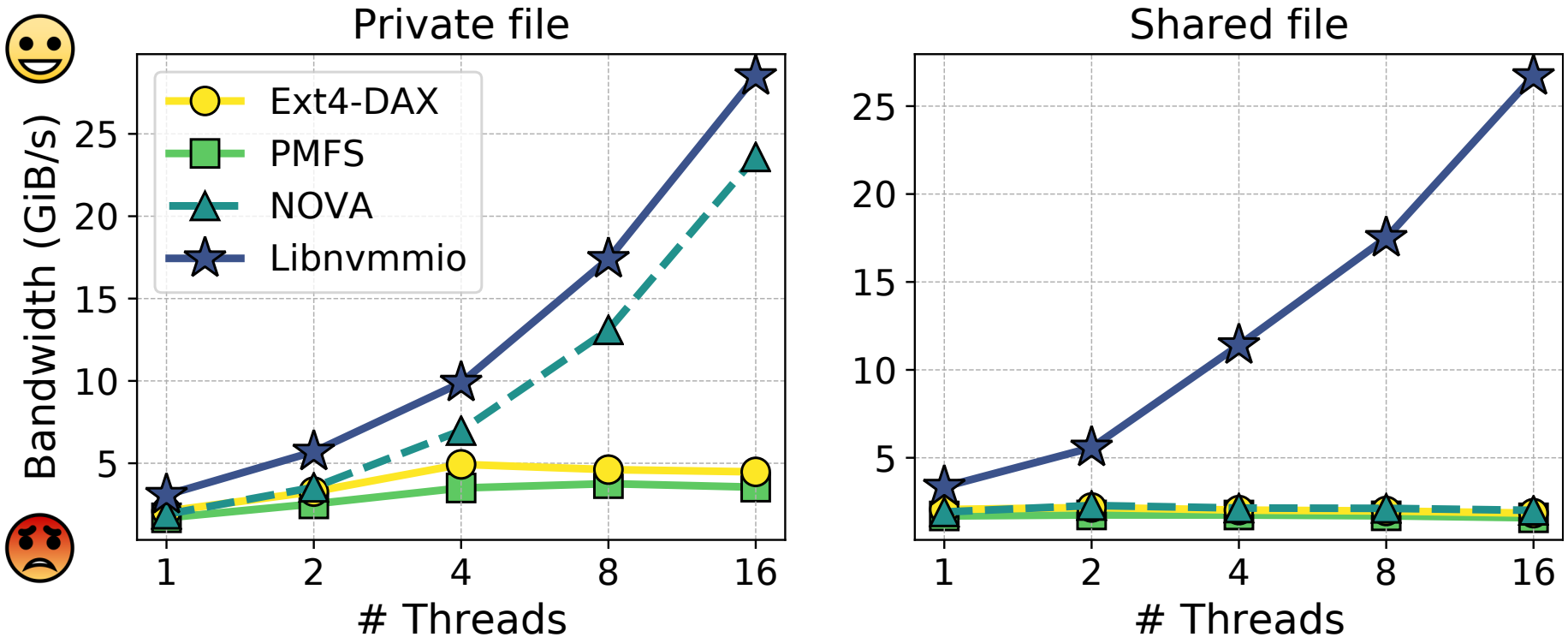
- A single thread, file size=4GB, block size=4KB, time=60s



FIO: Different Write Sizes

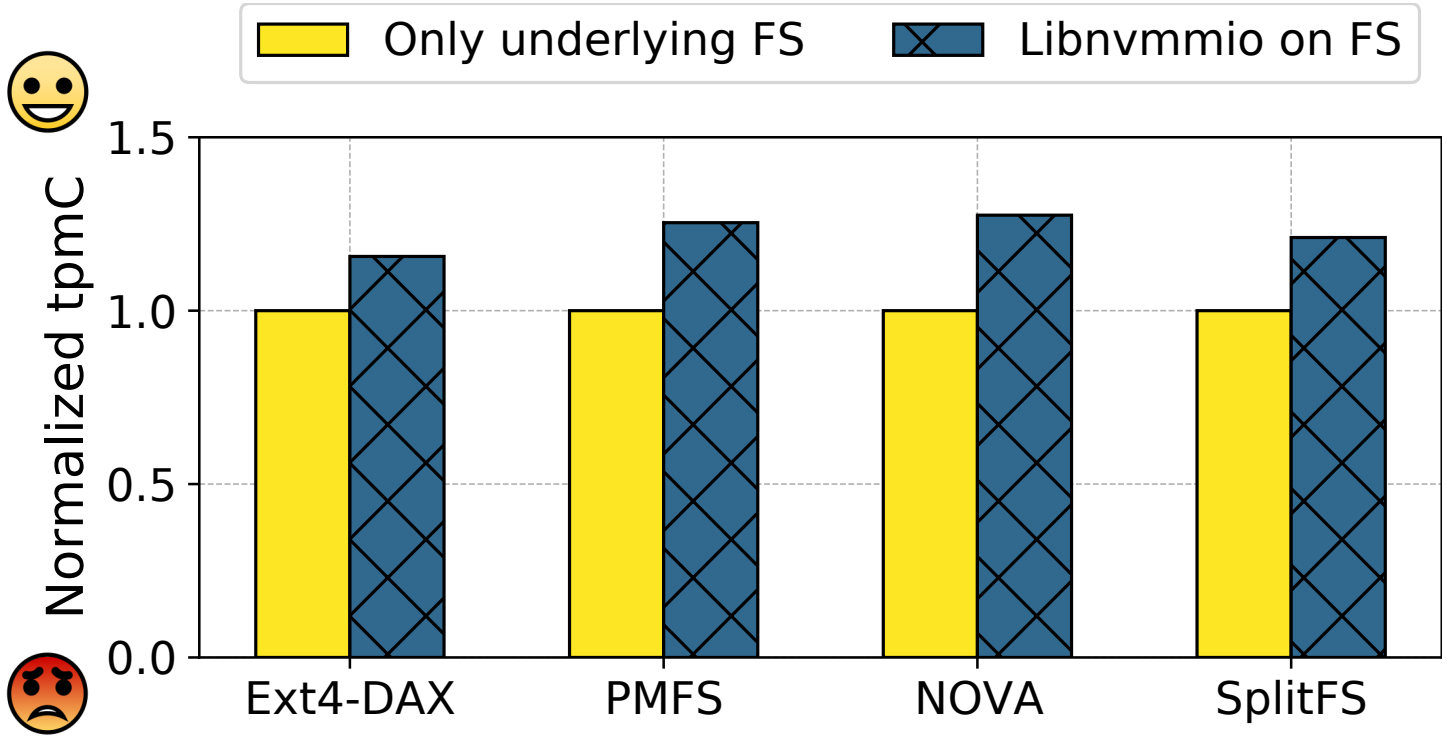


FIO: Random Write with Multithreads



TPC-C on SQLite

- Underlying FS with WAL, and Libnvmmio without WAL



SQLite WAL *vs.* Libnvmmio

- SQLite WAL
  - Design for block devices
  - Similar to REDO logging
  - Read both WAL and DB file
  - Only one writer at a time
  - Synchronous checkpointing
- Libnvmmio
  - Design for NVMM
  - Hybrid Logging
  - Read DB file (UNDO)
  - Concurrent writes
  - Background checkpointing
- Easily improve performance with Libnvmmio
  - Support any FS, Even FS that does not provide data-atomicity

Conclusion

- It is important to minimize SW overhead in NVMM systems
- Libnvmmio is a simple and practical solution
  - Reconstruct SW IO path
  - Run on any filesystem that provide DAX-mmap
- Low-latency, scalable IO while guaranteeing data-atomicity
  - 2.2x better throughput
  - 13x better scalability
- https://github.com/chjs/libnvmmio

QnA

chjs@skku.edu