Finding and Fixing Performance Pathologies in Persistent Memory Software Stacks

Jian Xu*, **Juno Kim***, Amirsaman Memaripour, Steven Swanson
UC San Diego

* denotes equal contribution

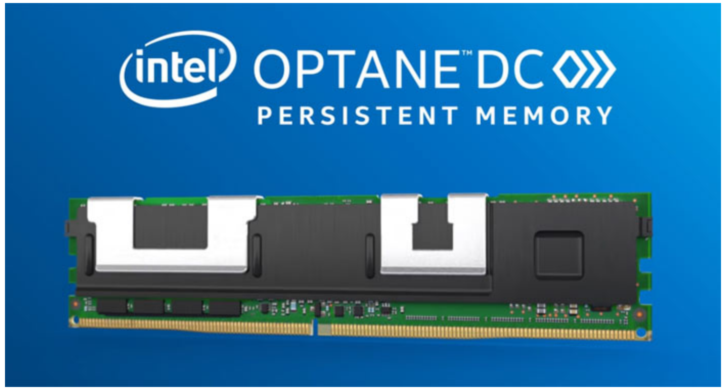
Persistent Memory

- New tier of memory
  - Low latency persistence (than SSD,HDD)
  - Large capacity (than DRAM)
- Intel Optane DC Persistent Memory
  - First scalable persistent memory
  - Re-evaluated some of our results on this device

Our paper

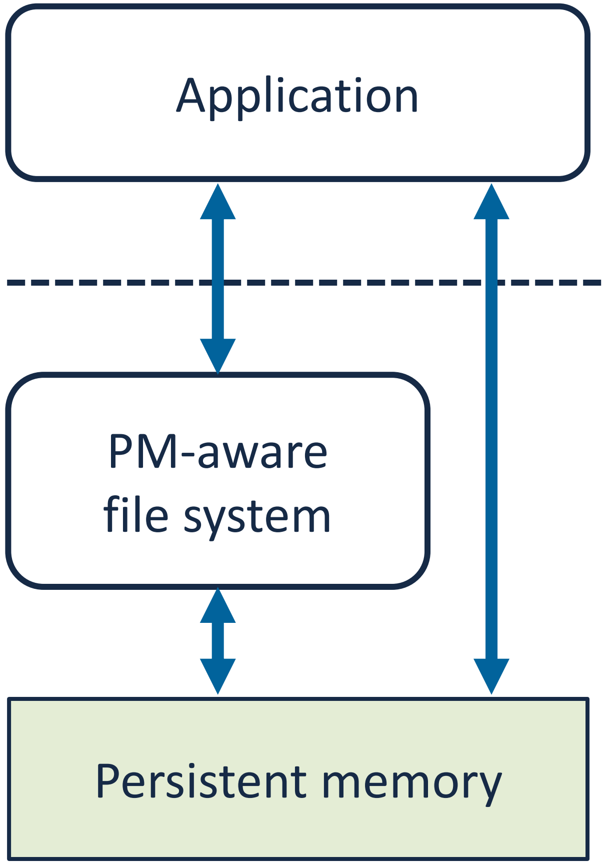
Battery-backed NVDIMM

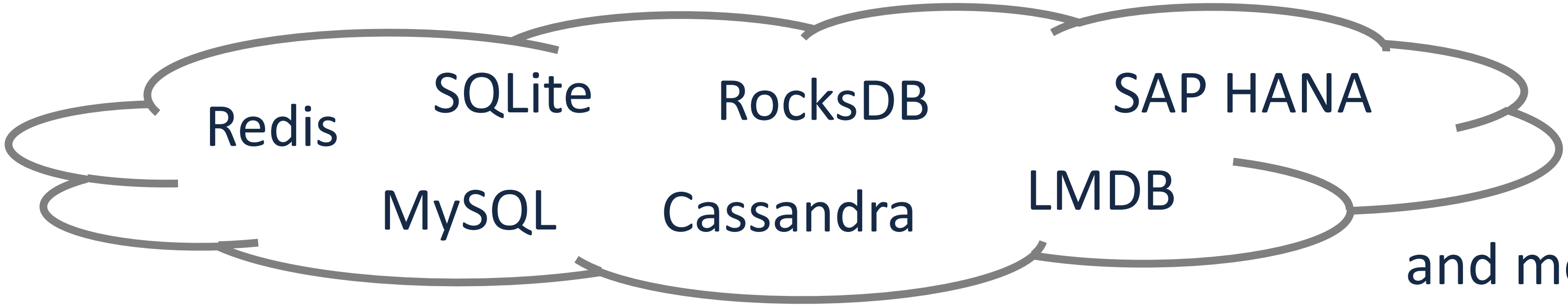
This talk



NVSL

Where are we now?





and more!

Legacy file systems
- XFS-DAX
- Ext4-DAX

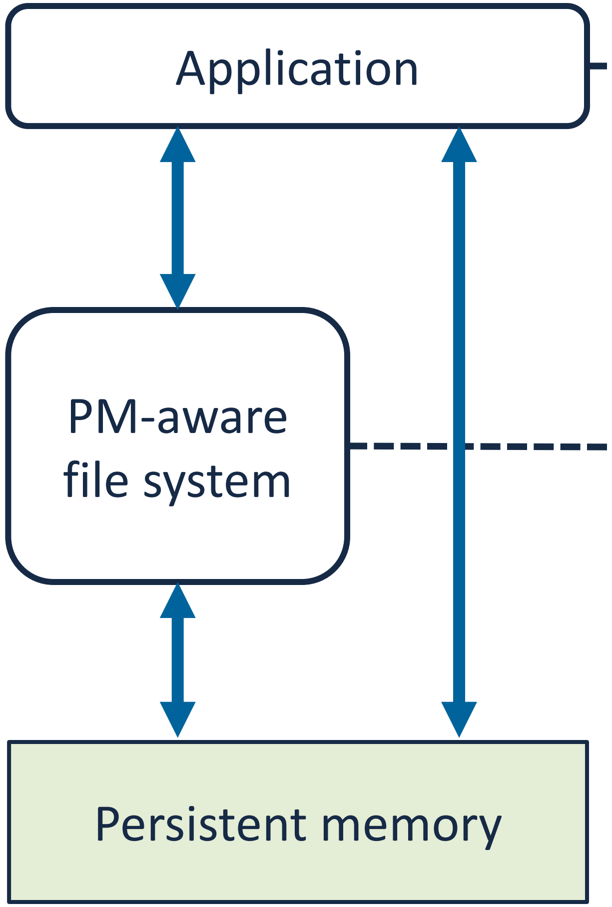
Custom file systems
- BPFS [SOSP'09]
- PMFS [Eurosys'14]
- NOVA [FAST'16]
- Strata [SOSP'17]

NVSL

Let’s see the whole picture

- Let’s fix the old codes
  - Legacy codes built for disk run slow on PM
- Let’s study the new trade-offs
  - What are the best ways to optimize software systems on PM?
  - What are the trade-offs? Complexity vs. Performance?
- **Our goal:** <u>fix urgent problems</u> and <u>provide best practices for optimization</u>.

Key questions

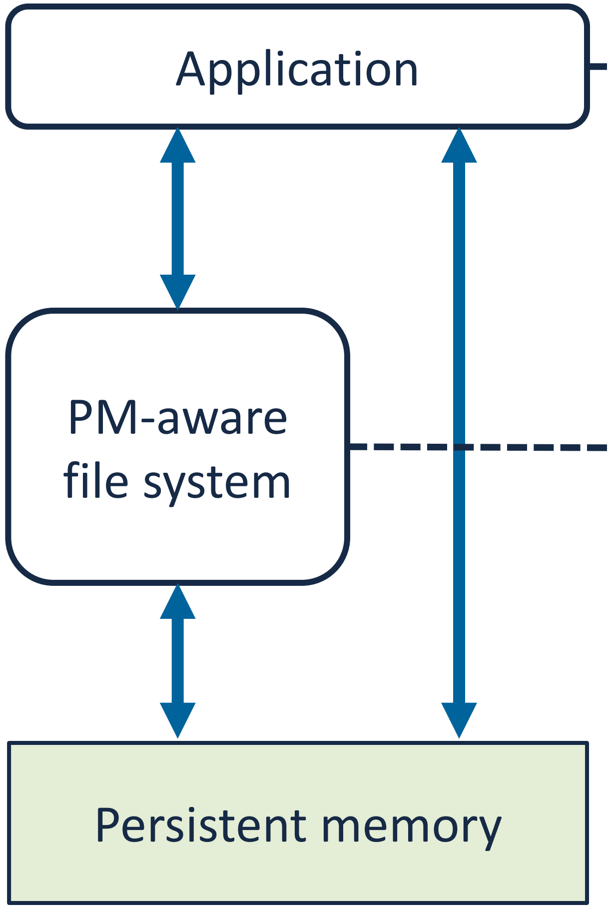


Which optimizations offer
the best complexity/performance trade-offs?

Are custom file systems worth it?

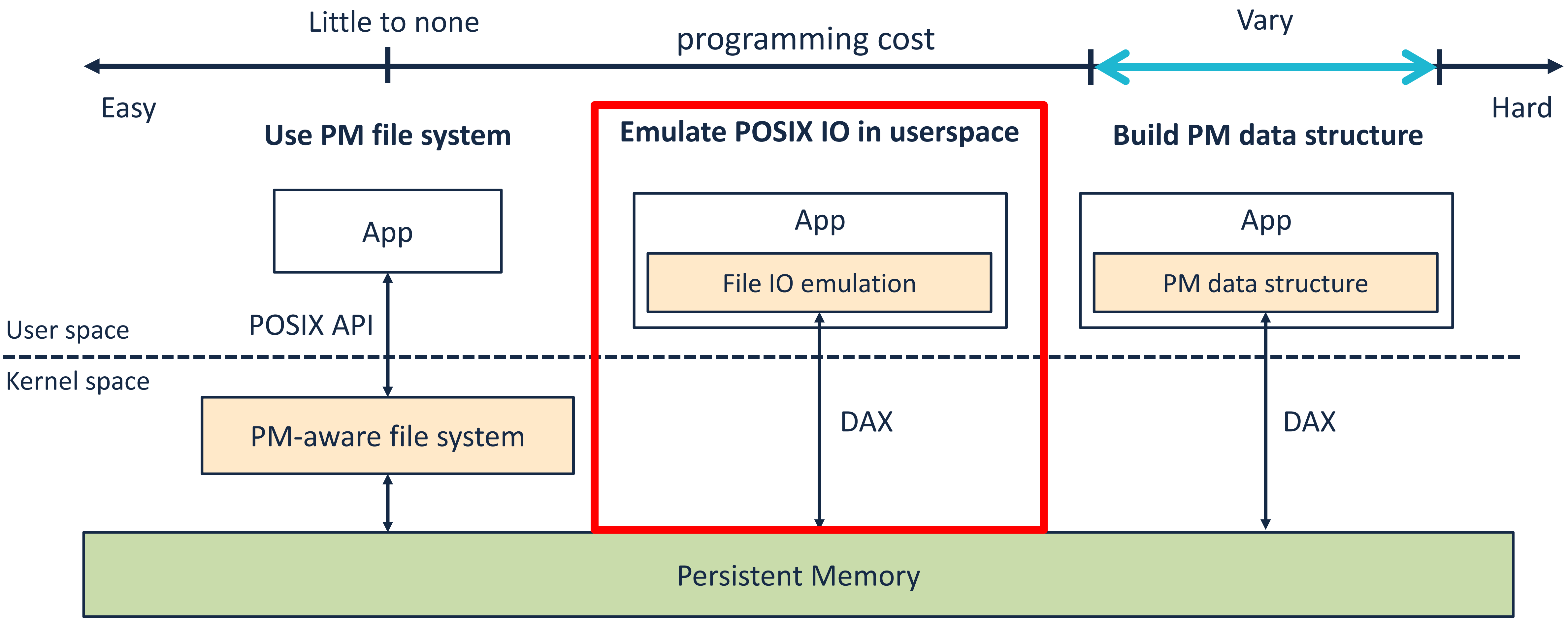
What bottlenecks remain?

Contributions



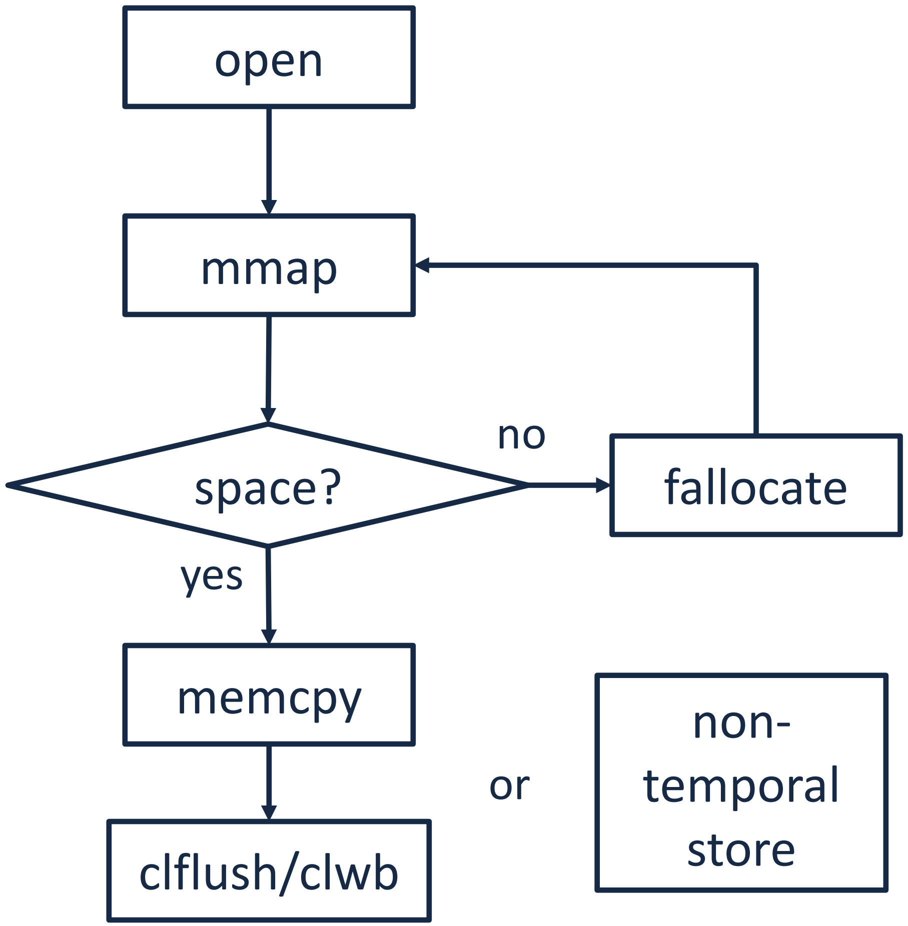
- Analyze a range of optimization techniques
- Show why custom file system is valuable
- Improve scalability for PM file systems

Candidate techniques for optimizing apps



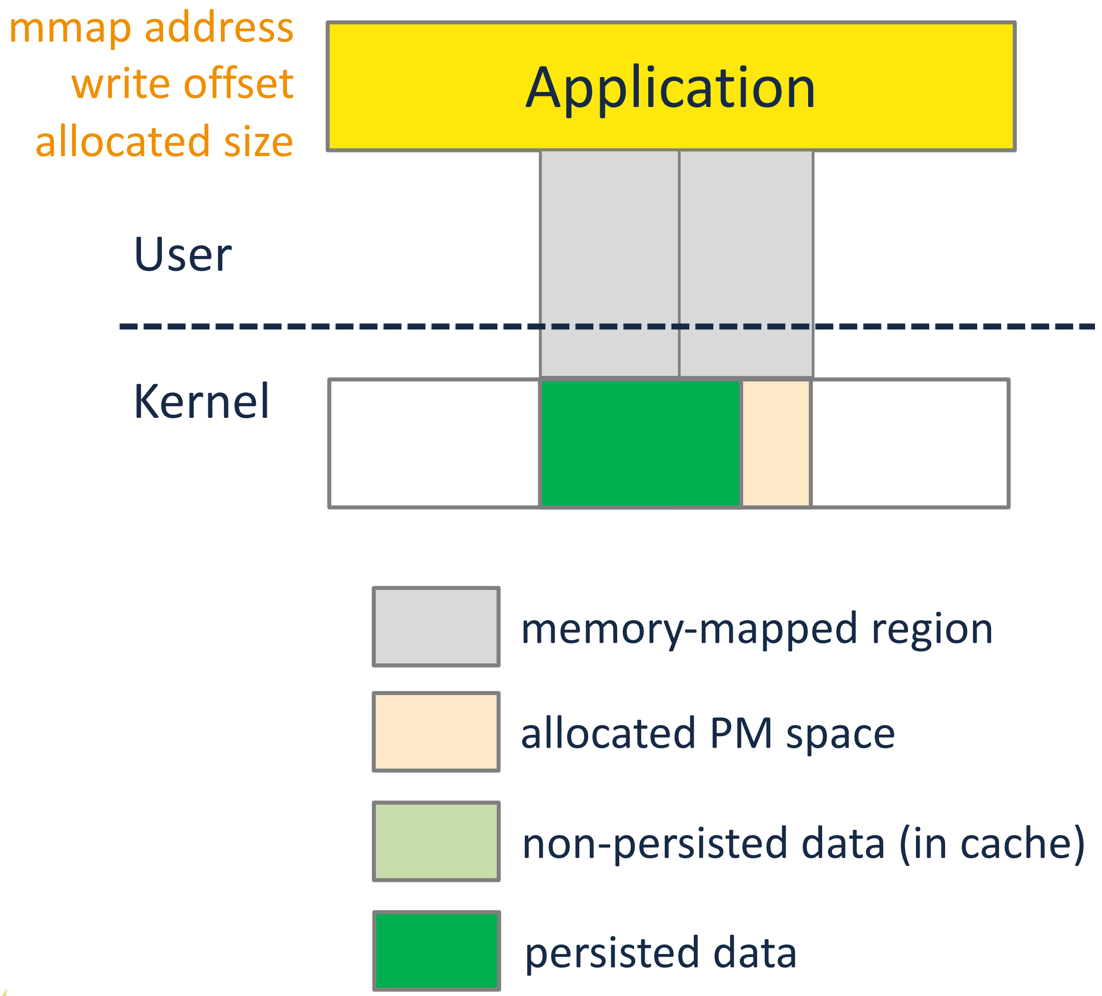
FLEX : FiLe Emulation with DAX

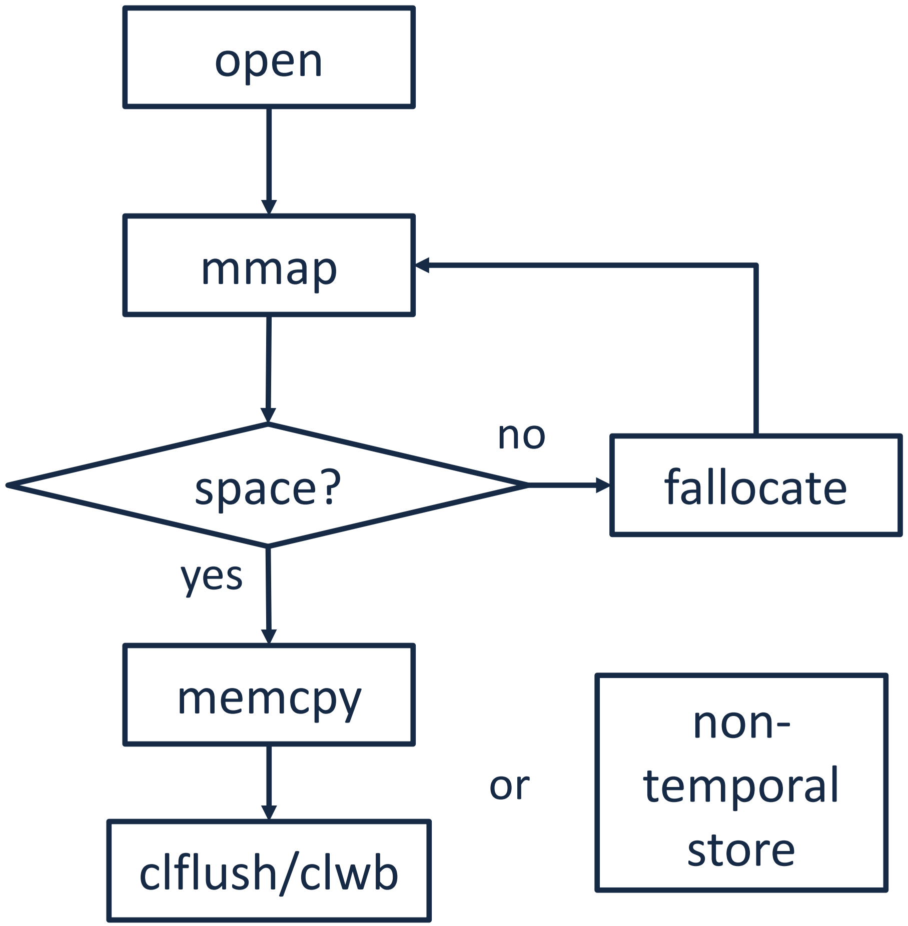
- Emulate POSIX IO in userspace with DAX
  - **open + mmap** a file
  - **memcpy + clflush/clwb** for write
  - **memcpy** for read
  - **fallocate + mmap** for extending file space
- Pros
  - Bypass file system overhead (e.g. journaling)
  - Amortize PM allocation cost by preallocation
- Cons
  - Guarantee only 8-byte atomicity



NVSL

FLEX append example





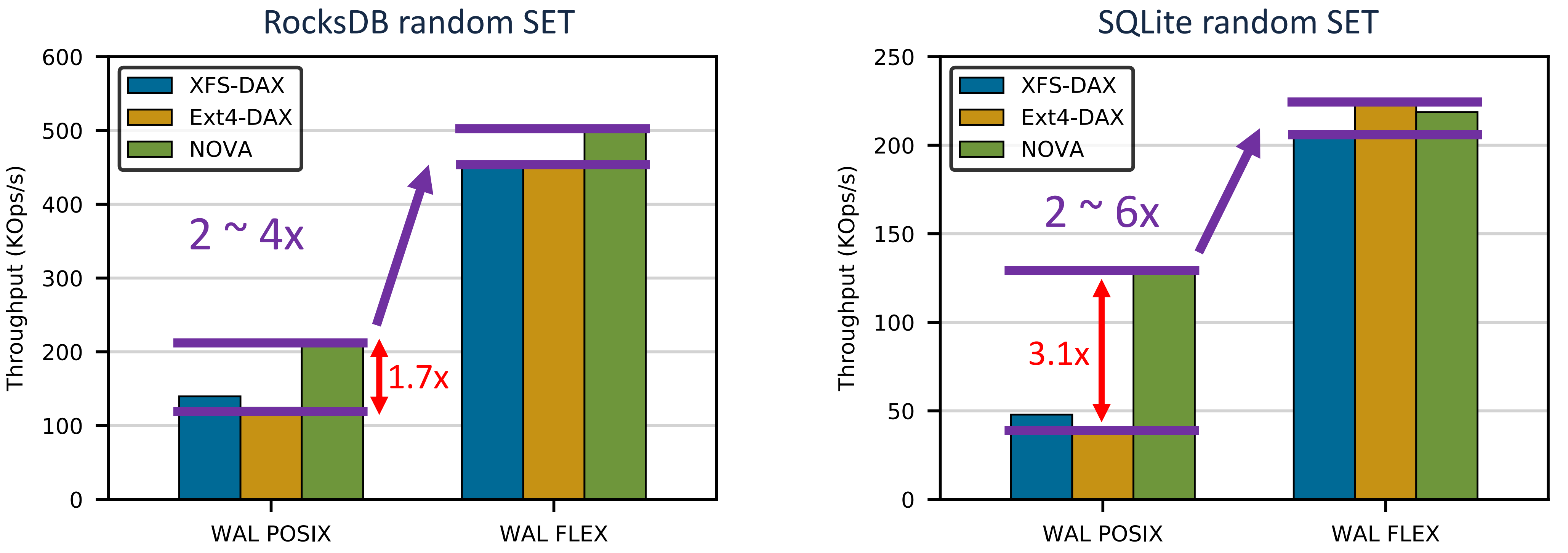
Applying FLEX to applications

- RocksDB, SQLite
  - Use file to implement Write-Ahead Logging (WAL) for consistency
- Most apps do NOT rely on the parts of POSIX that FLEX sacrifices [1]
  - Atomicity
  - File descriptor aliasing semantics
- Therefore, no logical change is required
  - RocksDB = 56 LOC, SQLite = 233 LOC

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

FLEX achieves substantial speedups

On Optane DC PM

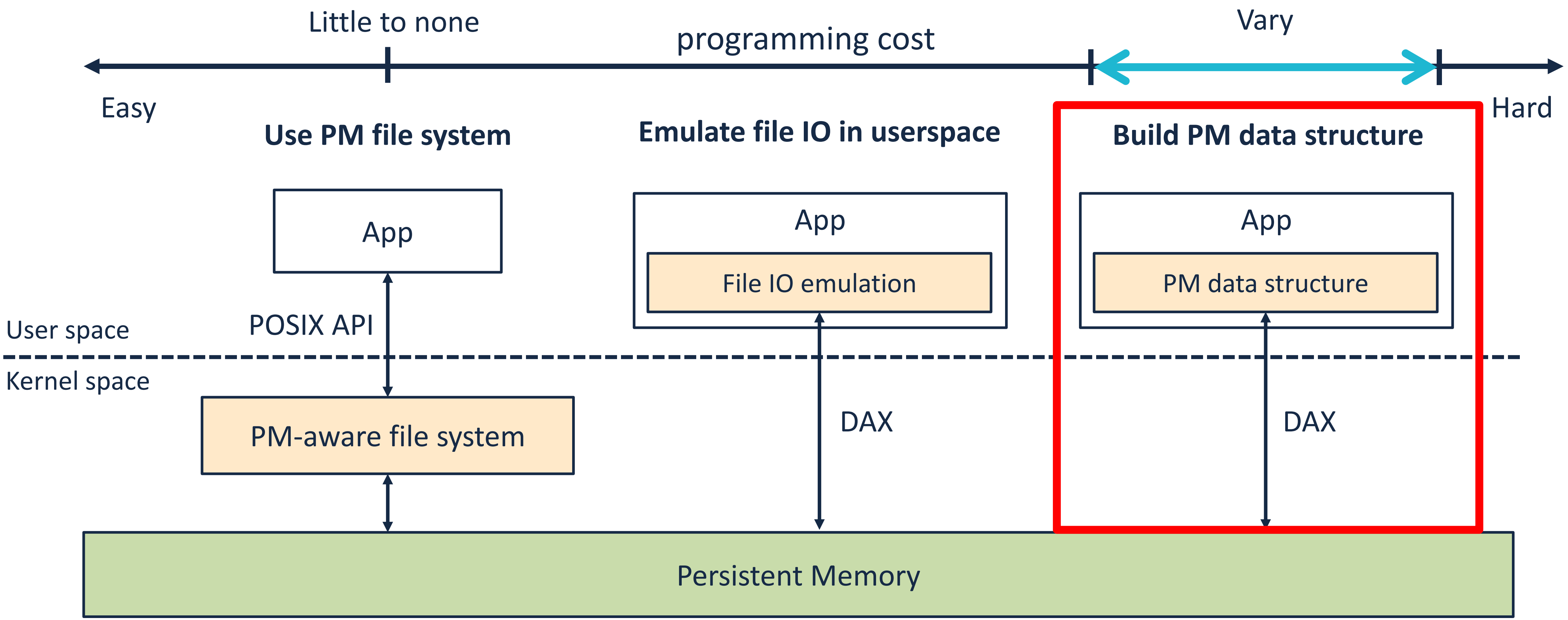


FLEX achieved 2 ~ 6x speedups over POSIX with simple changes.

FLEX reduces the gap between three file systems

NVSL

Let’s try a harder one

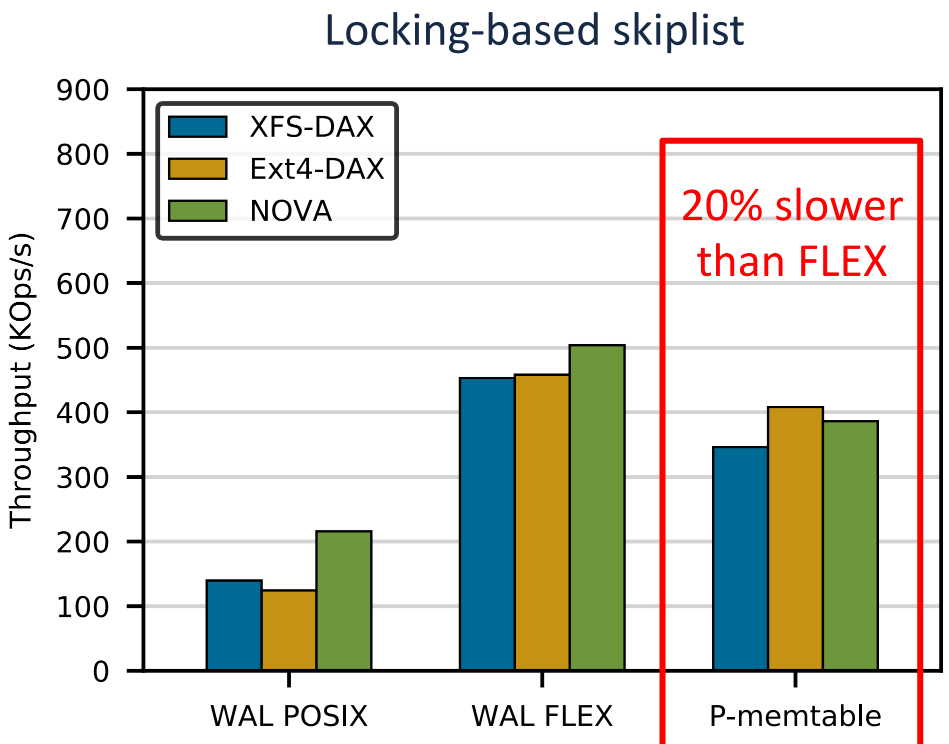


PM data structures

- Crash-consistent
  - No additional logging is required
- Difficult to build
  - Complex operations (e.g. B-tree split/merge, hash table resizing)
  - More challenging for concurrent data structures
- Recent progress
  - LSM-tree: NoveLSM [ATC'18], SLM-DB [FAST'19]
  - Hash-table: Level hashing [OSDI'18], CCEH [Fast'19]
  - B-tree: NV-Tree [FAST'15], FP-tree [SIGMOD'16]

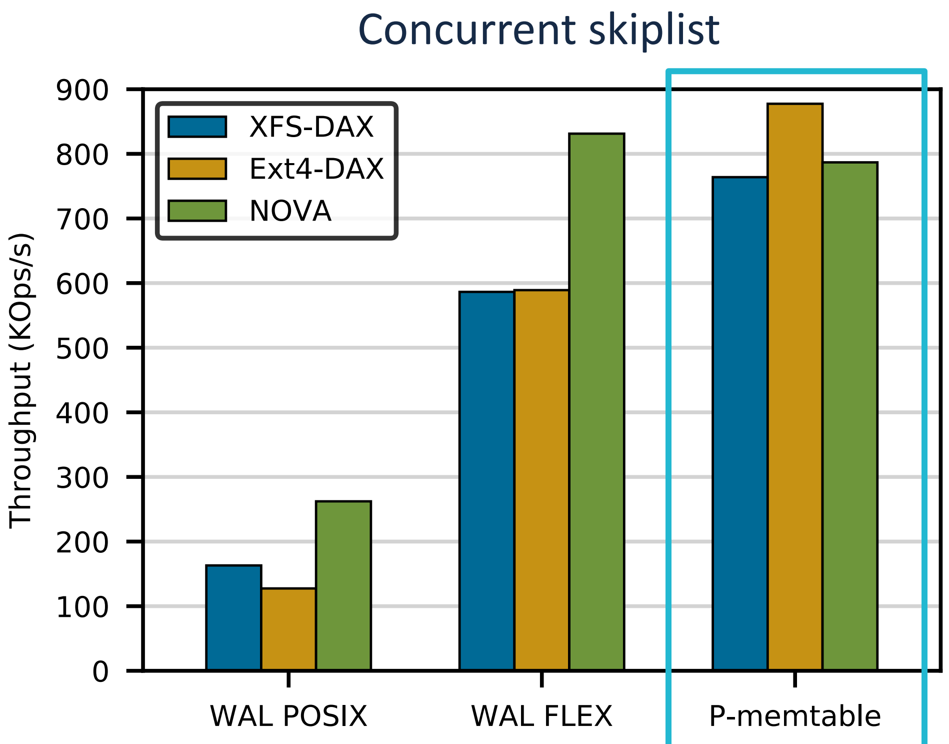
Persistent skiplist in RocksDB

On Optane DC PM



20% slower than FLEX

Modified lines: 56 380



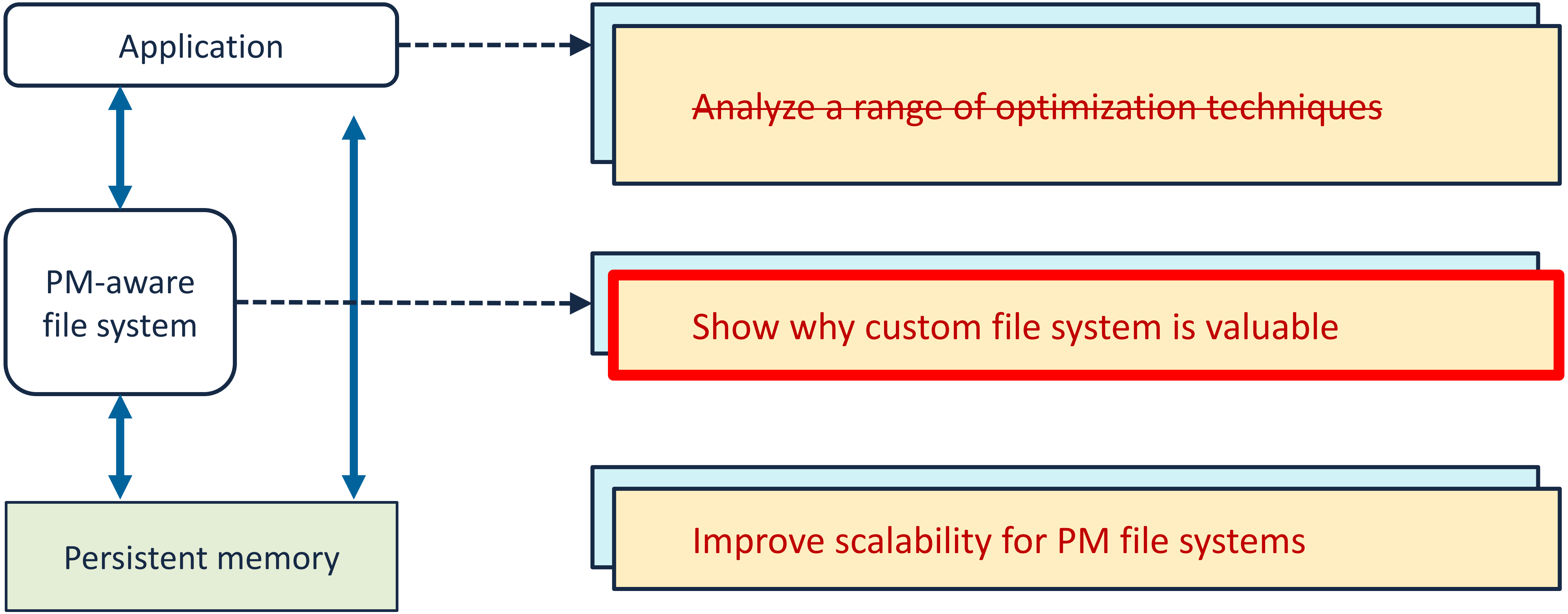
25% faster than FLEX

Modified lines: 56 380

Takeaway

- FLEX is a cost effective option for accelerating applications.
  - Some applications can do this easily.
- PM data structures can provide better performance but developers should carefully weigh the trade-offs.

Key questions



Why do we need another new file system?

- Legacy file systems already support PM access
  - XFS, EXT4 file systems are extended for PM → XFS-DAX, Ext4-DAX
- Can't we just improve them?
  - If we could get good performance out of one of these, we should!
- Let's try optimizing Ext4-DAX!

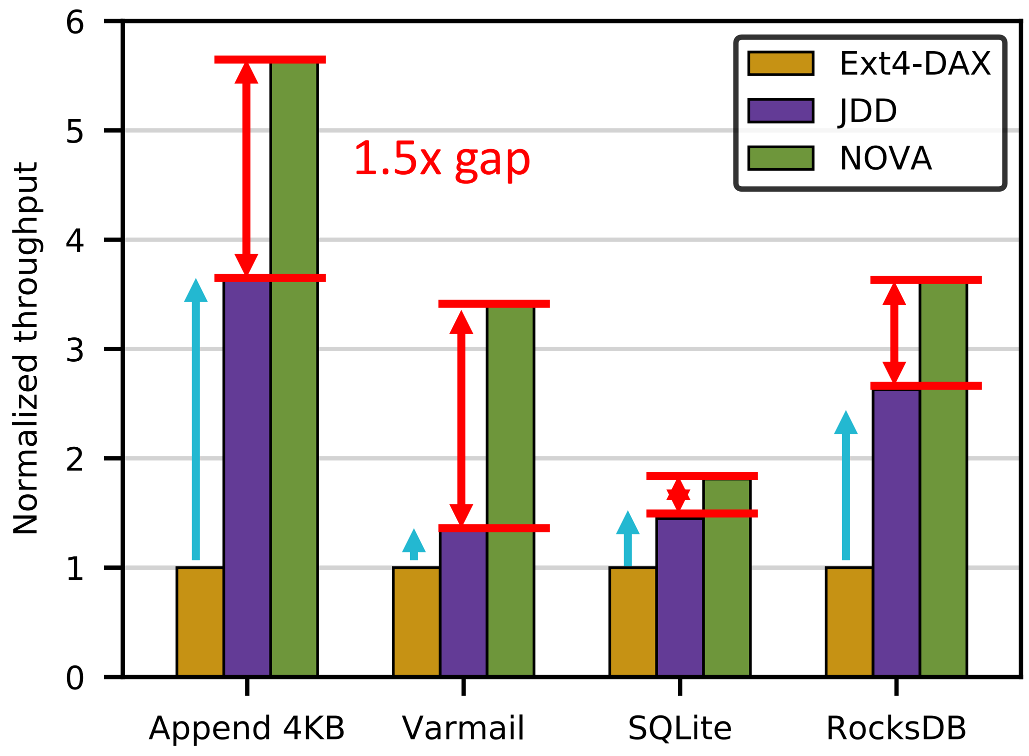
Fine-grained journaling for Ext4-DAX

- Key overhead: **block-based legacy journaling device (JBD2)**
  - Write amplification: E.g. 4KB data append → 36KB writes to file/journal
  - Global journaling area → No concurrency

- Our solution: **Journaling DAX Device (JDD)**
  - Journals individual metadata fields → No write amplification
  - Pre-allocates per-CPU journaling area → Good scalability
  - Undo logging → Simplified commit mechanism (e.g. no checkpointing)

JDD performance

- Compare with Ext4-DAX, NOVA
- Run four benchmarks
  - Append 4KB
  - Filebench varmail
  - SQLite (the same before)
  - RocksDB (the same before)
- Result
  - Faster than Ext4-DAX by 2.3x
  - NOVA is still 1.5x faster.



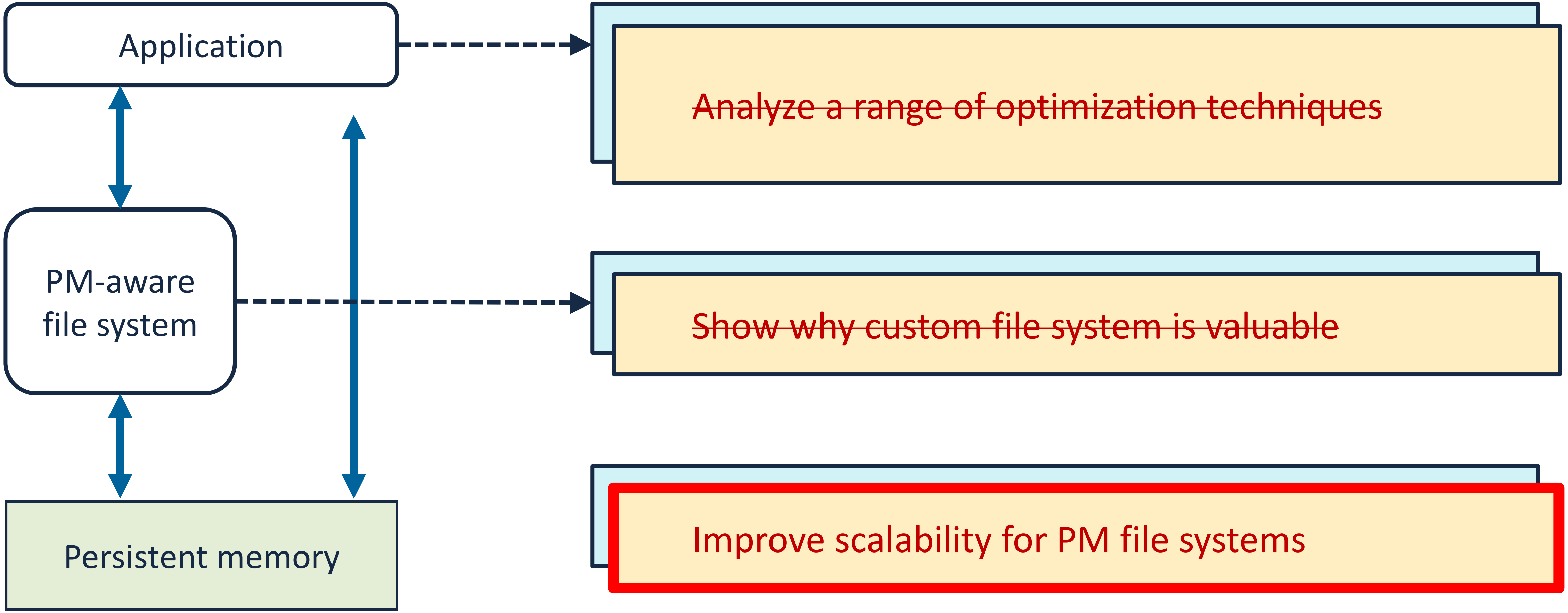
NVSL

Can we fill the gap further?

- “Disk first”
  - Ext4-DAX shares codebase with disk-oriented Ext4
  - Disruptive changes are not likely to happen
  - Further optimizations would make Ext4 a less-good disk-based file system.

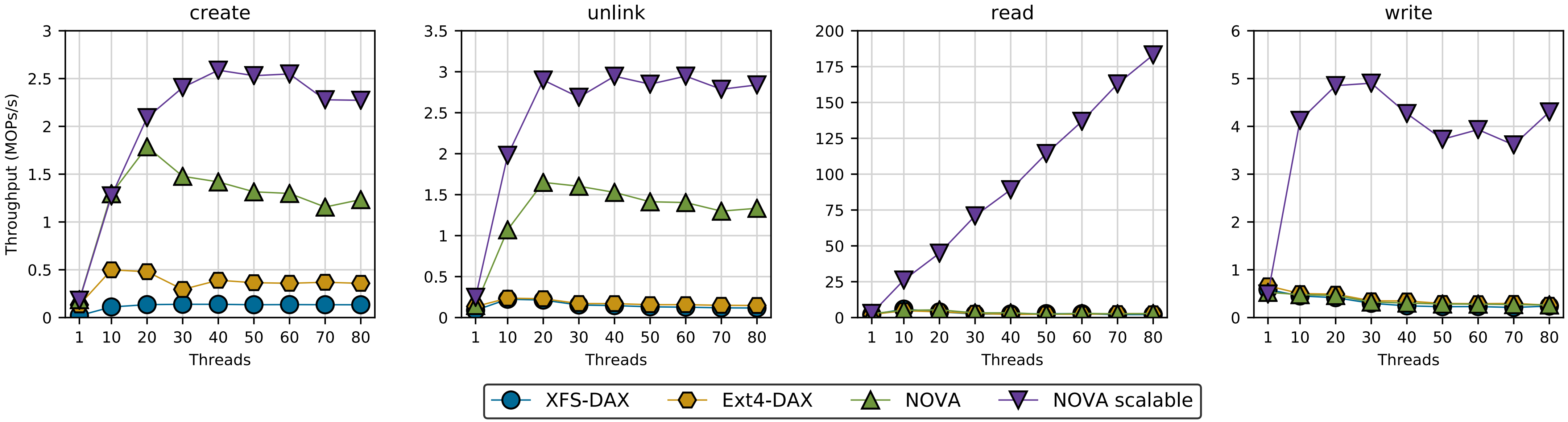
- We do actually need a custom file system for PM!

Key questions



Poor scalability by Virtual File System

- **Bottleneck**: Global inode structure, per-inode locking
- **Solution**: Per-CPU inode structure, fine-grained locking
- See our paper for details

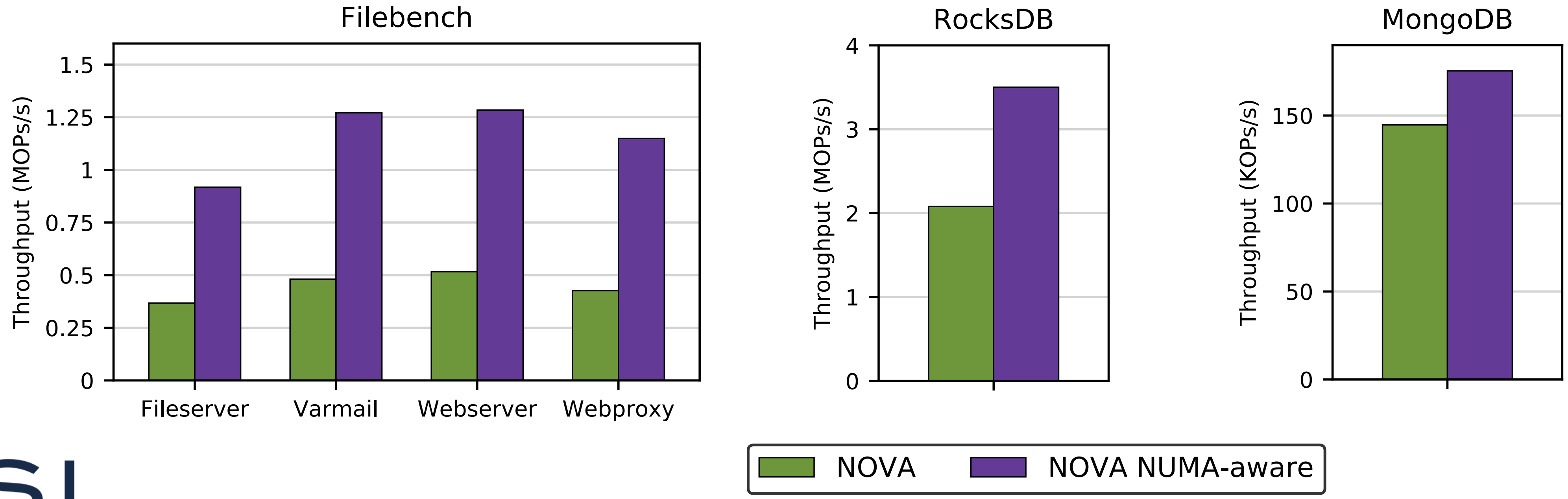


NVSL

[1] Min et al, Understanding Manycore Scalability of File Systems, ATC'16

Better scalability with NUMA-aware file access

- Enabled NUMA-aware file access in NOVA
  - Added simple interface for querying/setting NUMA location per file
  - Achieved 1.2 – 2.6x better throughput
- See our paper for details



NVSL

Conclusion

- FLEX is a cost-effective app optimization technique.
- PM data structures can provide better performance but developers should carefully weigh the trade-offs.
- Custom file system provides better performance and legacy file systems are unlikely to close the gap.
- Memory-centric optimizations (e.g. NUMA) are now applicable (and profitable) for file.

Thank you! Questions?

NVSL