NOVA-Fortis: A Fault-Tolerant Non-Volatile Main Memory File System

Jian Andiry Xu, **Lu Zhang**, Amirsaman Memaripour, Akshatha Gangadharaiah, Amit Borase, Tamires Brito Da Silva, Andy Rudoff (Intel), Steven Swanson

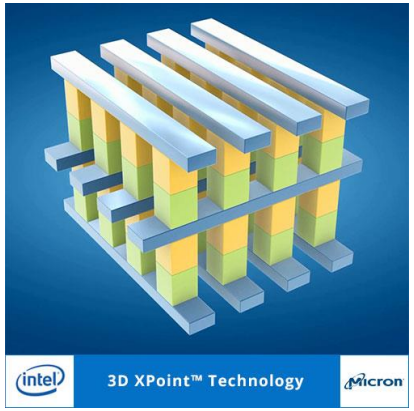
Non-Volatile Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego

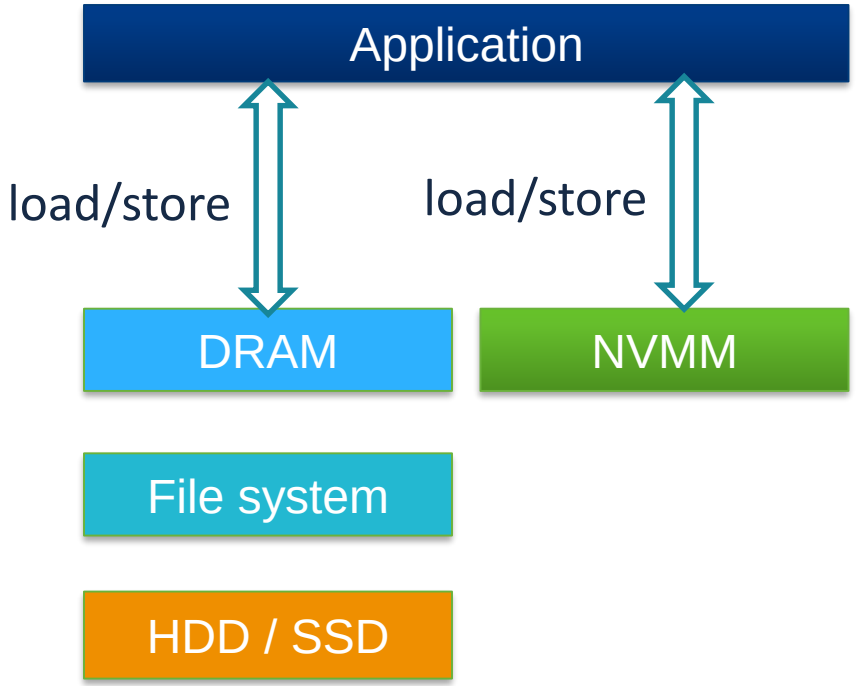
NVSL

UCSDCSE Computer Science and Engineering

Non-volatile Memory and DAX

- Non-volatile main memory (NVMM)
  - PCM, STT-RAM, ReRAM, 3D XPoint technology
  - Reside on memory bus, load/store interface

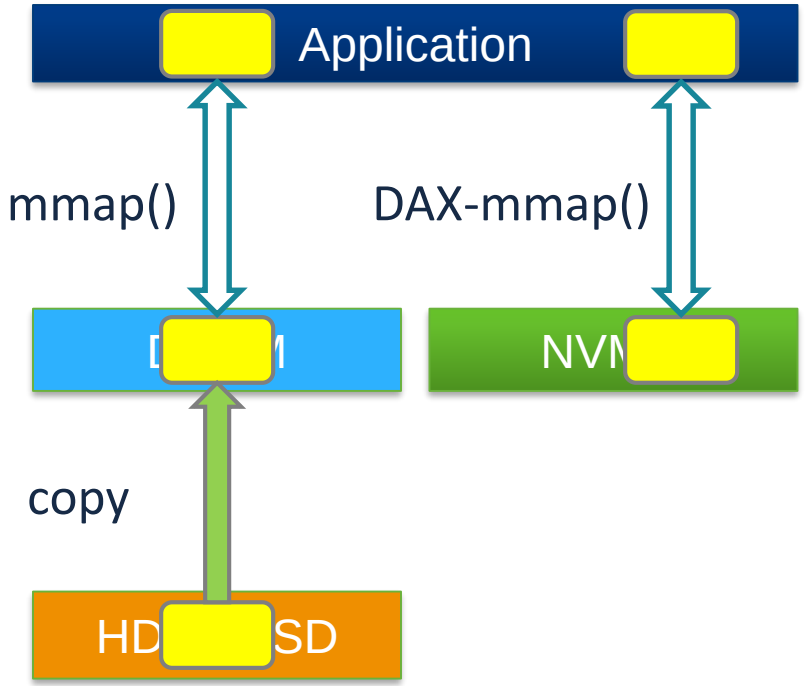




NVSL

Non-volatile Memory and DAX

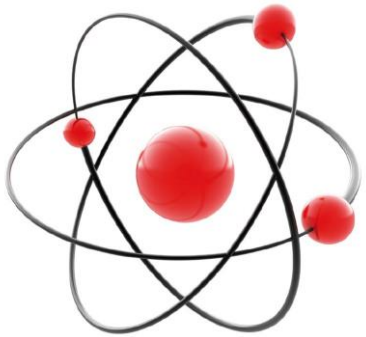
- Non-volatile main memory (NVMM)
  - PCM, STT-RAM, ReRAM, 3D XPoint technology
  - Reside on memory bus, load/store interface
- Direct Access (DAX)
  - DAX file I/O bypasses the page cache
  - DAX-mmap() maps NVMM pages to application address space directly and bypasses file system
  - “Killer app”



NVSL

Application expectations on NVMM File System

POSIX I/O



Atomicity

Fault Tolerance

Direct Access

Speed

ext4 xfs BtrFS F2FS

POSIX I/O

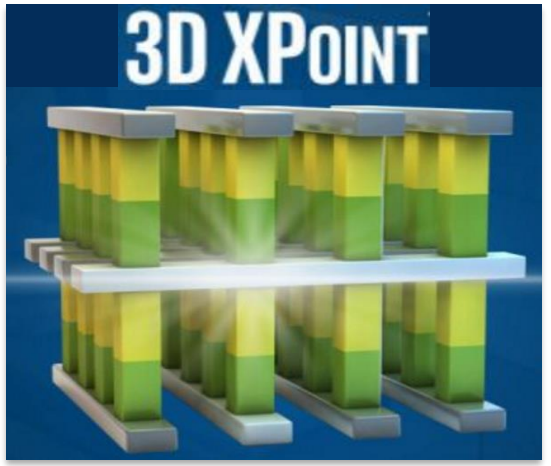
Atomicity

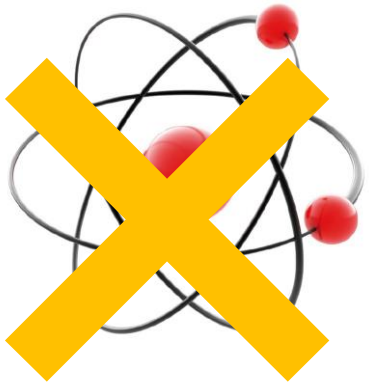
Fault Tolerance

Direct Access

Speed

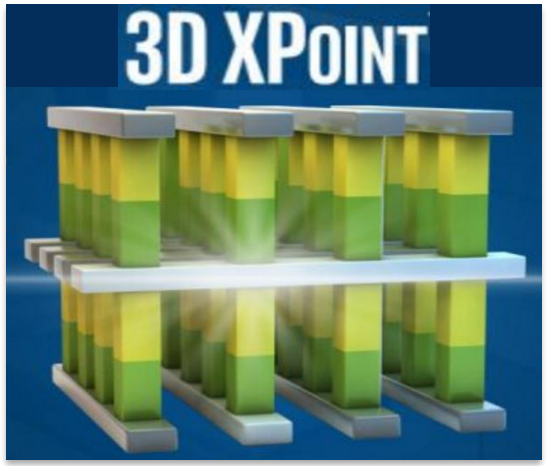
PMFS ext4-DAX xfs-DAX

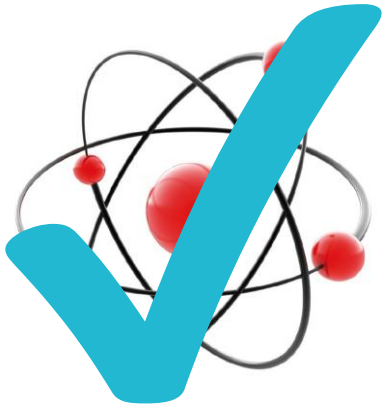




Strata

SOSP ’17

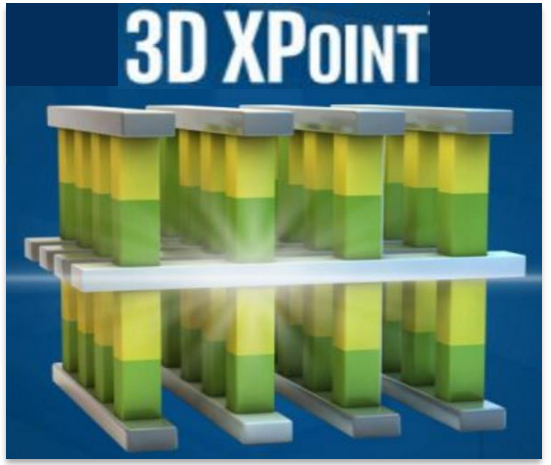




NOVA

FAST ’16


AGIGA

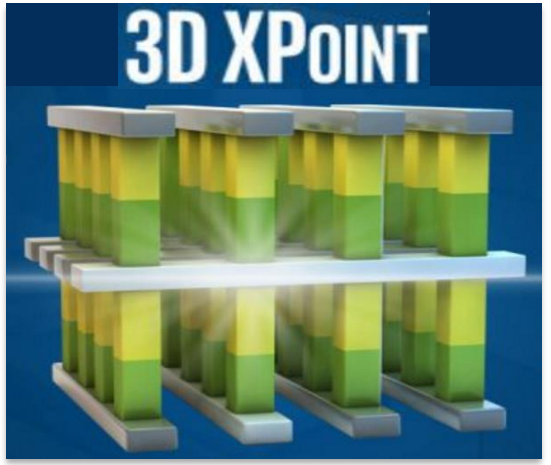

3D XPOINT


ORACLE
DATABASE
Microsoft
SQL Server
TERADATA
MySQL
BASE
H2
Access
HIVE
SQLite
HyperSQL


DAX

NOVA-Fortis


AGIGA


3D XPOINT

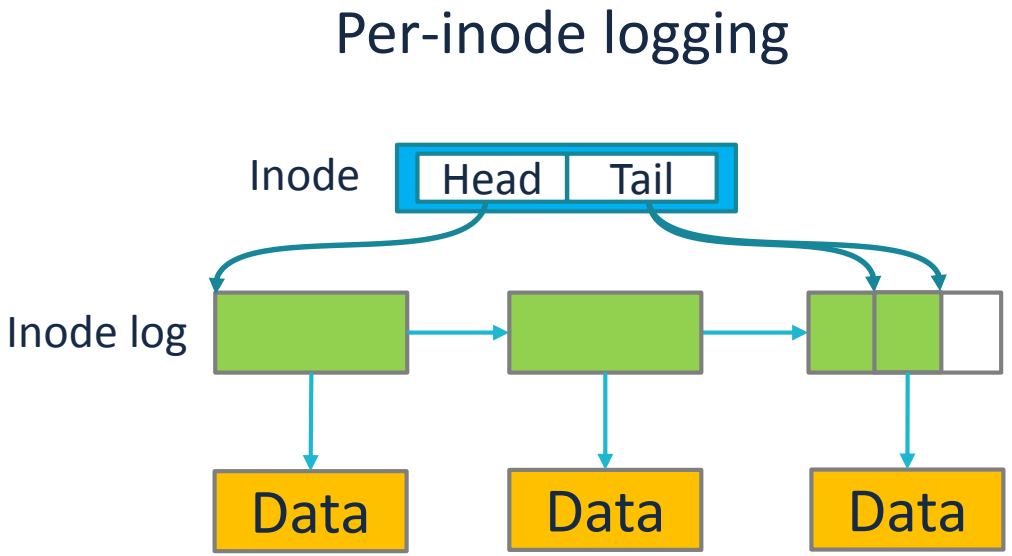

ORACLE
DATABASE
Microsoft
SQL Server
TERADATA
MySQL
H2
Access
HIVE
SQLite
HyperSQL


DAX

Challenges

NOVA: Log-structured FS for NVMM

- Per-inode logging
  - High concurrency
  - Parallel recovery
- High scalability
  - Per-core allocator, journal and inode table
- Atomicity
  - Logging for single inode update
  - Journaling for update across logs
  - Copy-on-Write for file data



NVSL

Jian Xu and Steven Swanson, NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories, FAST '16.

Snapshot

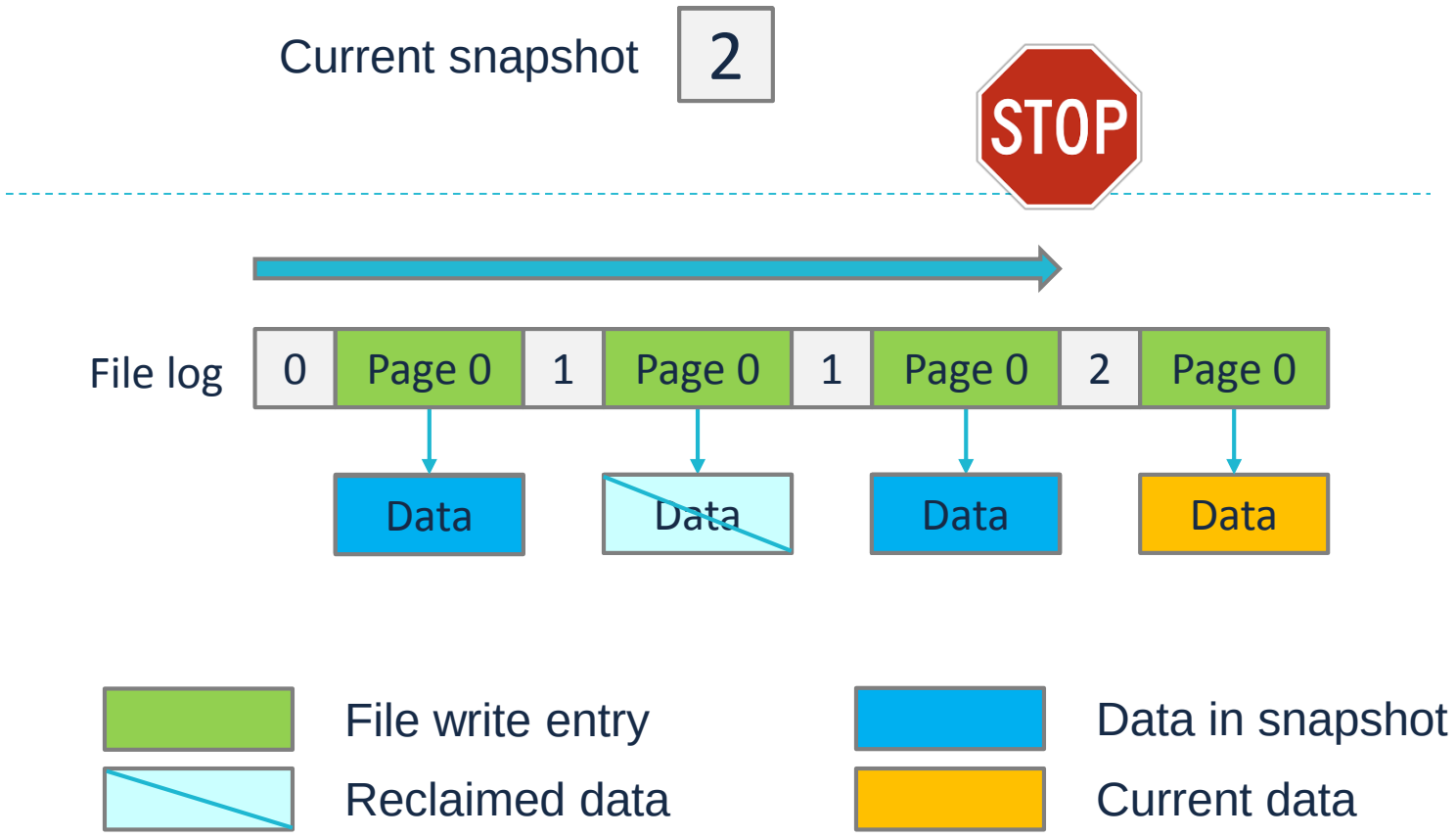
Snapshot support

- Snapshot is essential for file system backup
- Widely used in enterprise file systems
  - ZFS, Btrfs, WAFL

- Snapshot is not available with DAX file systems

NVSL

Snapshot for normal file I/O




```
write(0, 4K);
take_snapshot();
write(0, 4K);
write(0, 4K);
take_snapshot();
write(0, 4K);
recover_snapshot(1);
```

Memory Ordering With DAX-mmap()

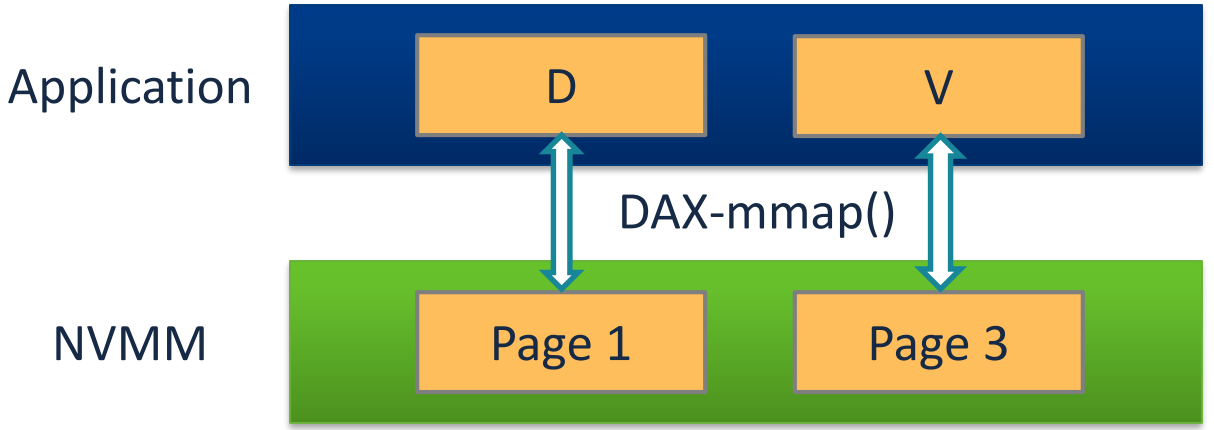
```
D = 42;
Fence();
V = True;
```

D	V	Valid
?	False	✓
42	False	✓
42	True	✓
?	True	X

- Recovery invariant: if V == True, then D is valid

Memory Ordering With DAX-mmap()

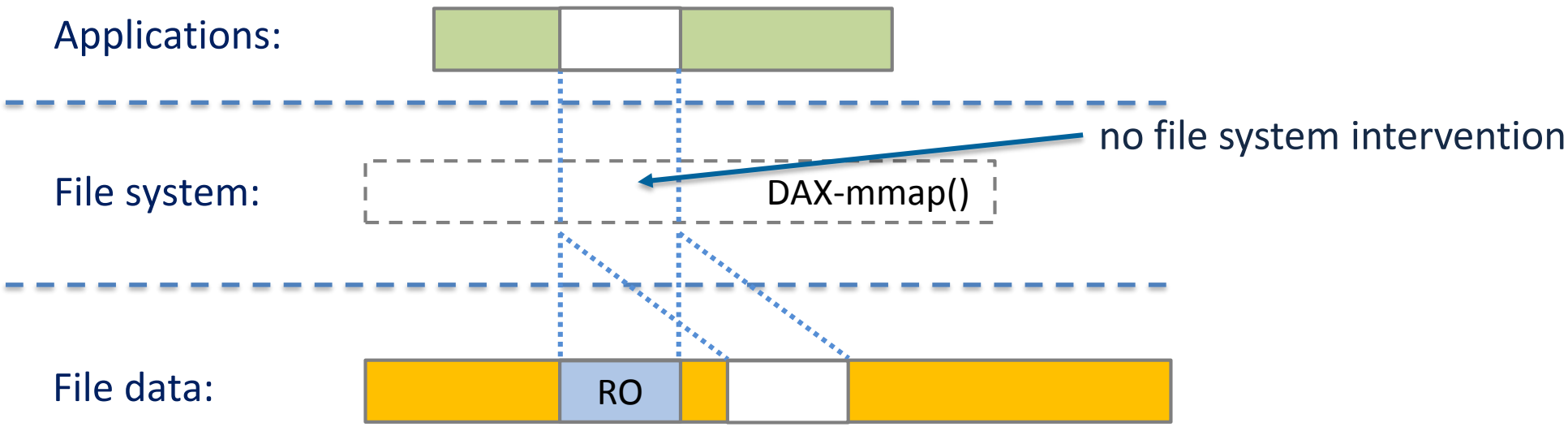
```
D = 42;
Fence();
V = True;
```



- Recovery invariant: if V == True, then D is valid
- D and V live in two pages of a mmap()'d region.

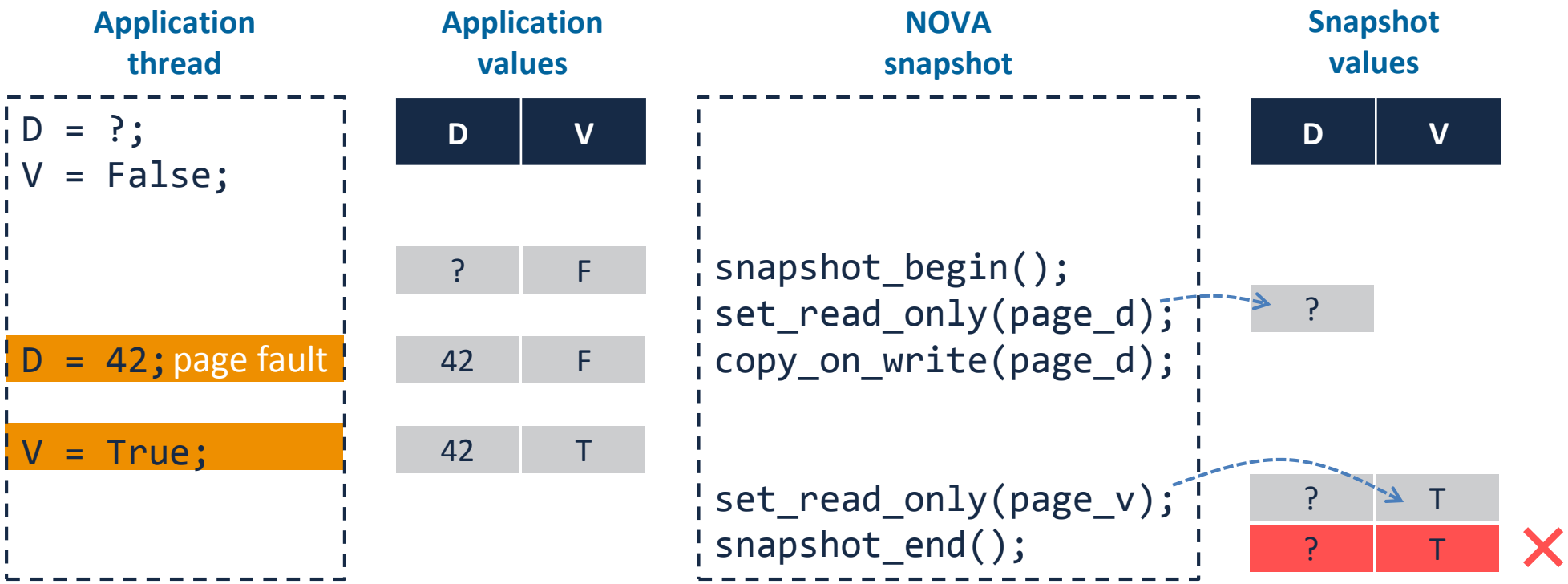
DAX Snapshot: Idea

- Set pages read-only, then copy-on-write



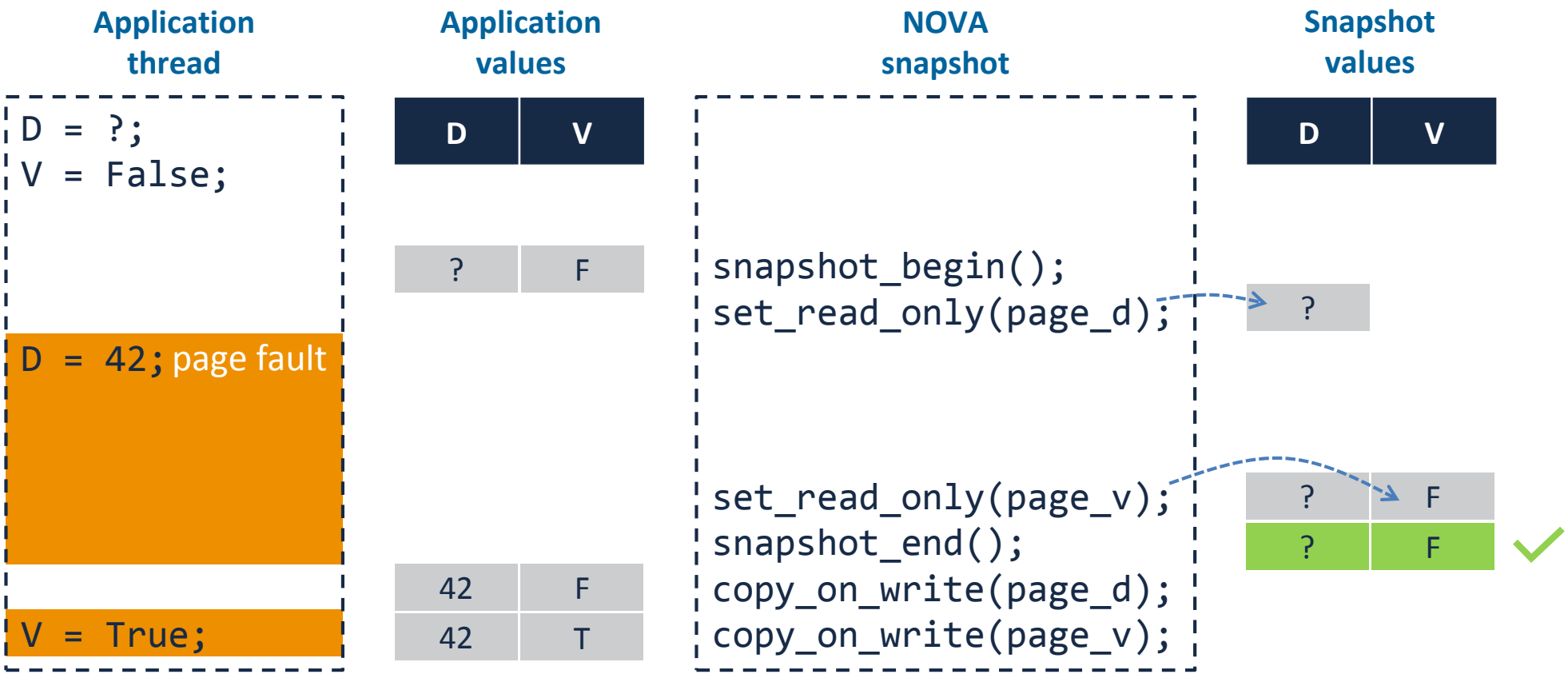
DAX Snapshot: Incorrect implementation

- Application invariant: if V is True, then D is valid



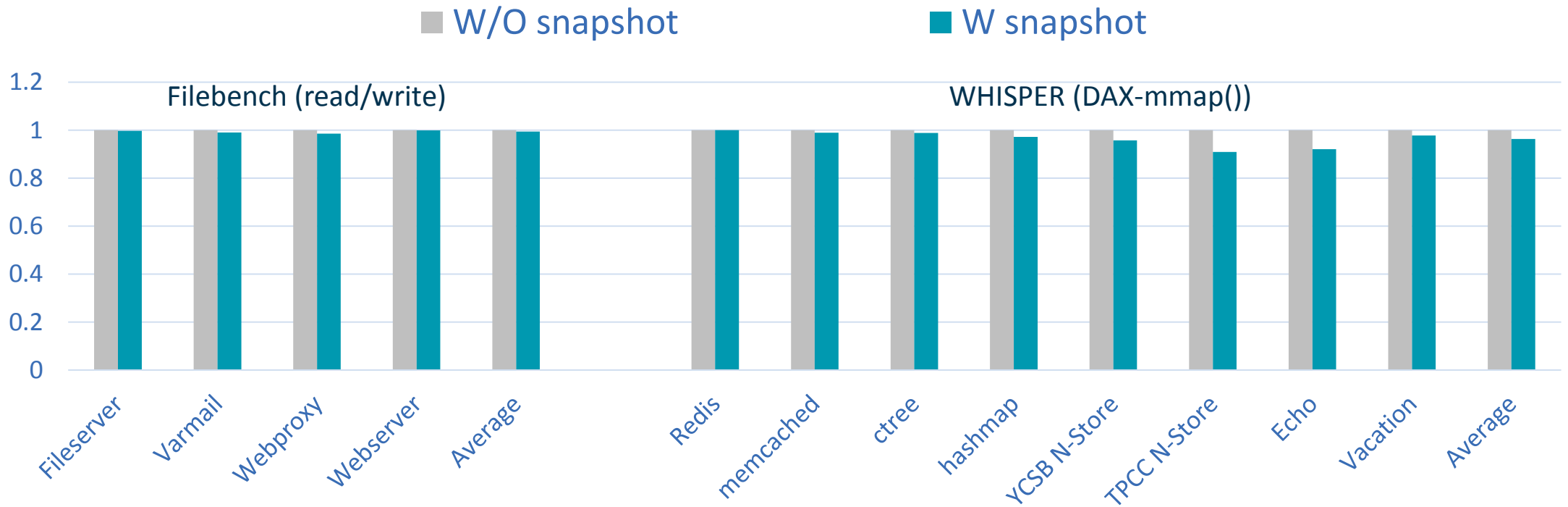
DAX Snapshot: Correct implementation

- Delay CoW page faults completion until all pages are read-only



Performance impact of snapshots

- Normal execution vs. taking snapshots every 10s
  - Negligible performance loss through read()/write()
  - Average performance loss 3.7% through mmap()



Protecting Metadata and Data

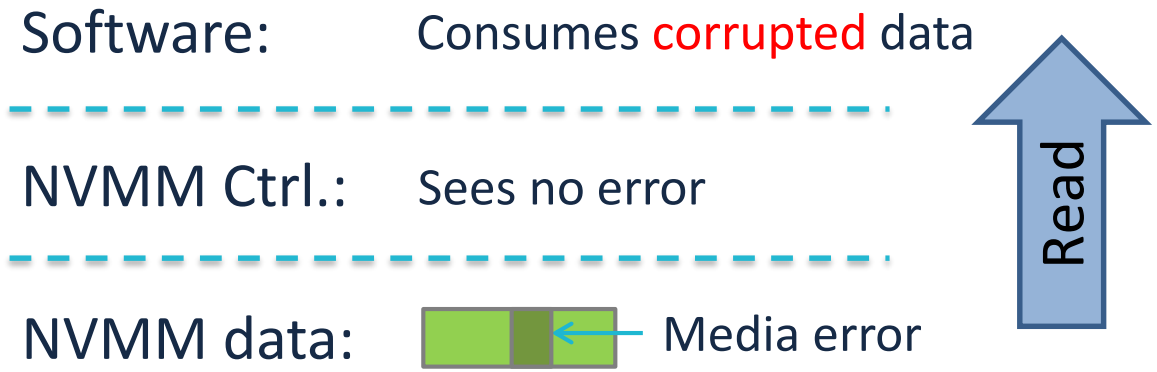
NVMM Failure Modes

- Detectable errors
  - Media errors detected by NVMM controller
  - Raises Machine Check Exception (MCE)
- Undetectable errors
  - Media errors not detected by NVMM controller
  - Software scribbles

NVSL

NVMM Failure Modes

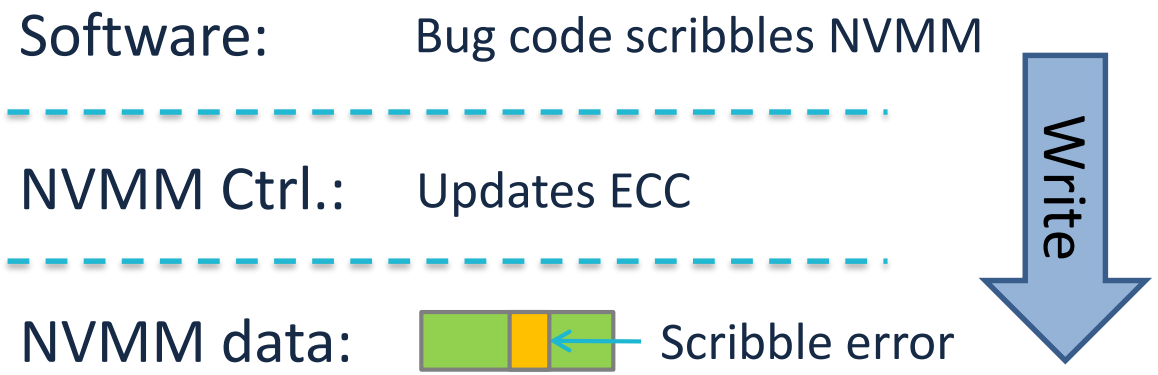
- Detectable errors
  - Media errors detected by NVMM controller
  - Raises Machine Check Exception (MCE)
- Undetectable errors
  - Media errors not detected by NVMM controller
  - Software scribbles



NVSL

NVMM Failure Modes

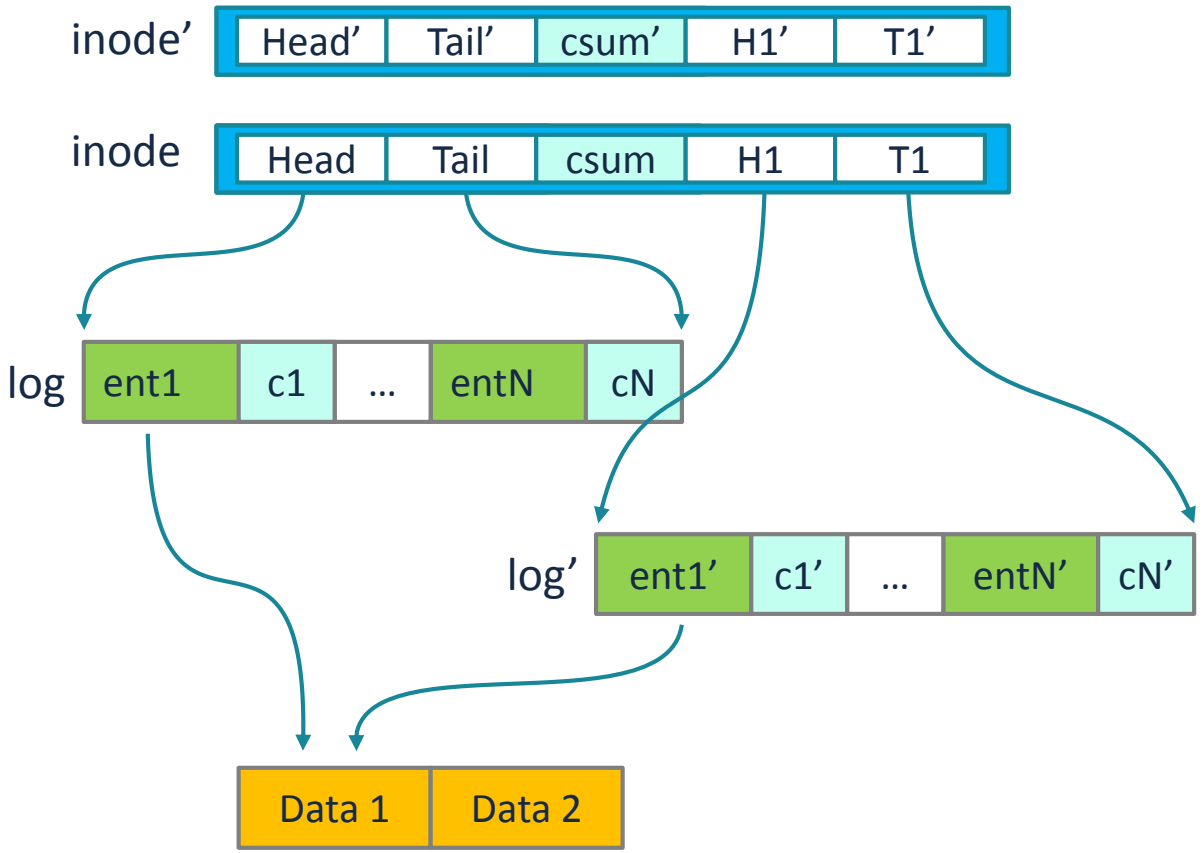
- Detectable errors
  - Media errors detected by NVMM controller
  - Raises Machine Check Exception (MCE)
- Undetectable errors
  - Media errors not detected by NVMM controller
  - Software scribbles



NVSL

NOVA-Fortis Metadata Protection

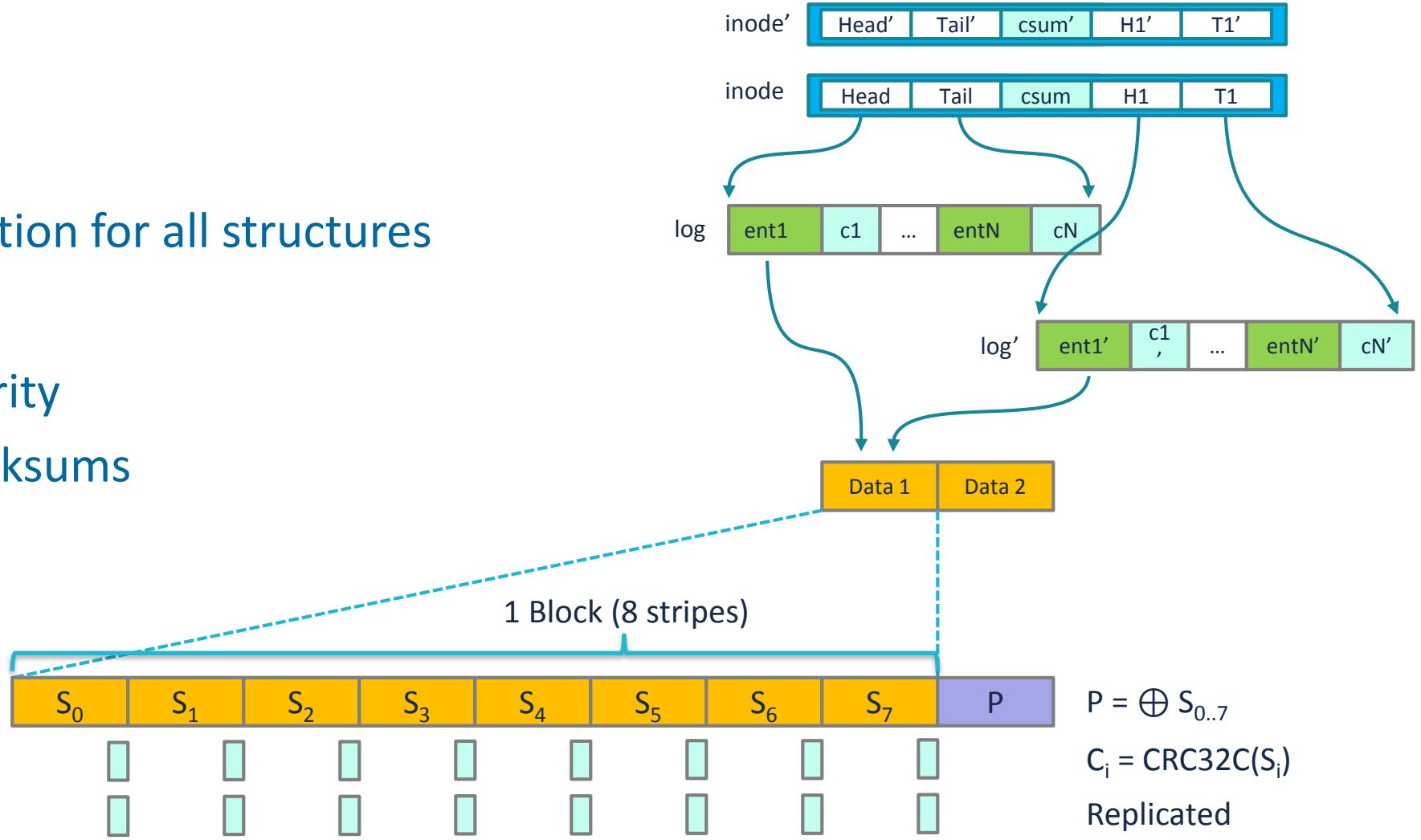
- Detection
  - CRC32 checksums in all structures
  - Use memcpy_mcsafe() to catch MCEs
- Correction
  - Replicate all metadata: inodes, logs, superblock, etc.
  - Tick-tock: persist primary before updating replica



NVSL

NOVA-Fortis Data Protection

- Metadata
  - CRC32 + replication for all structures
- Data
  - RAID-4 style parity
  - Replicated checksums



File data protection with DAX-mmap

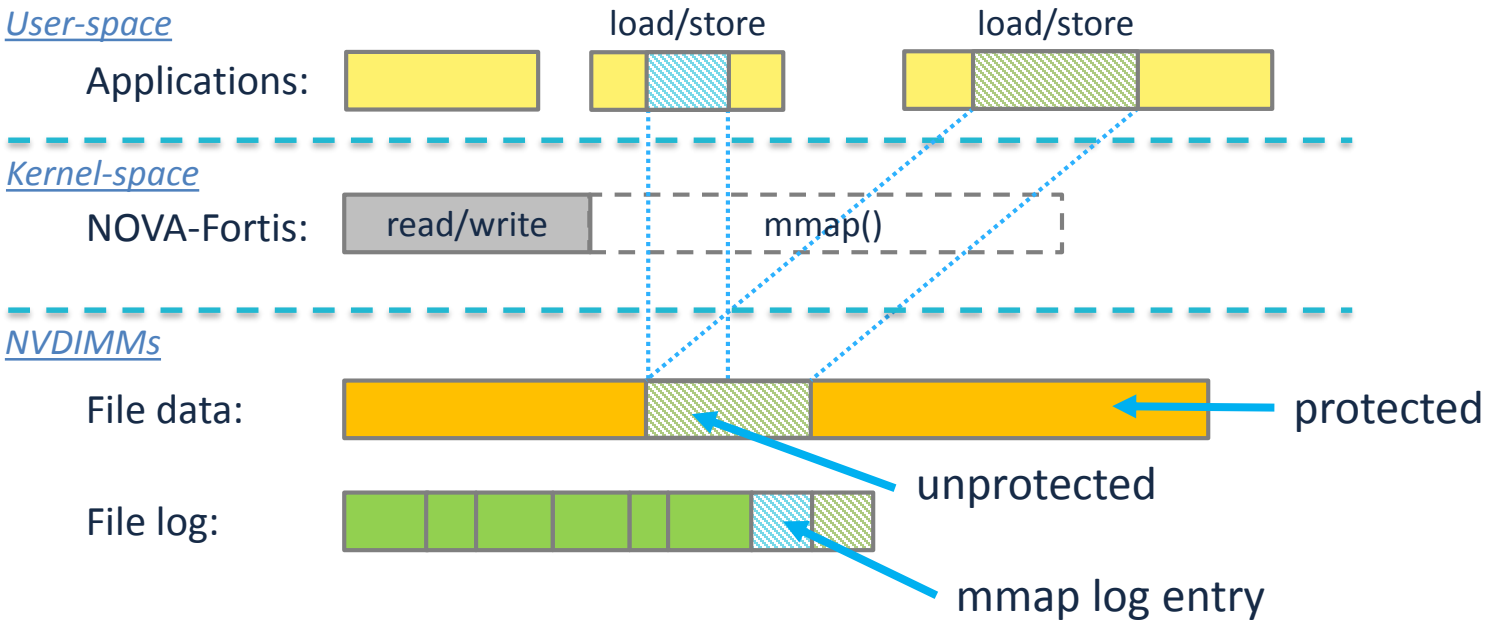
- Stores are invisible to the file systems
- The file systems cannot protect mmap'ed data
- NOVA-Fortis' data protection contract:

NOVA-Fortis protects pages from media errors and scribbles iff they are not mmap()'d for writing.

NVSL

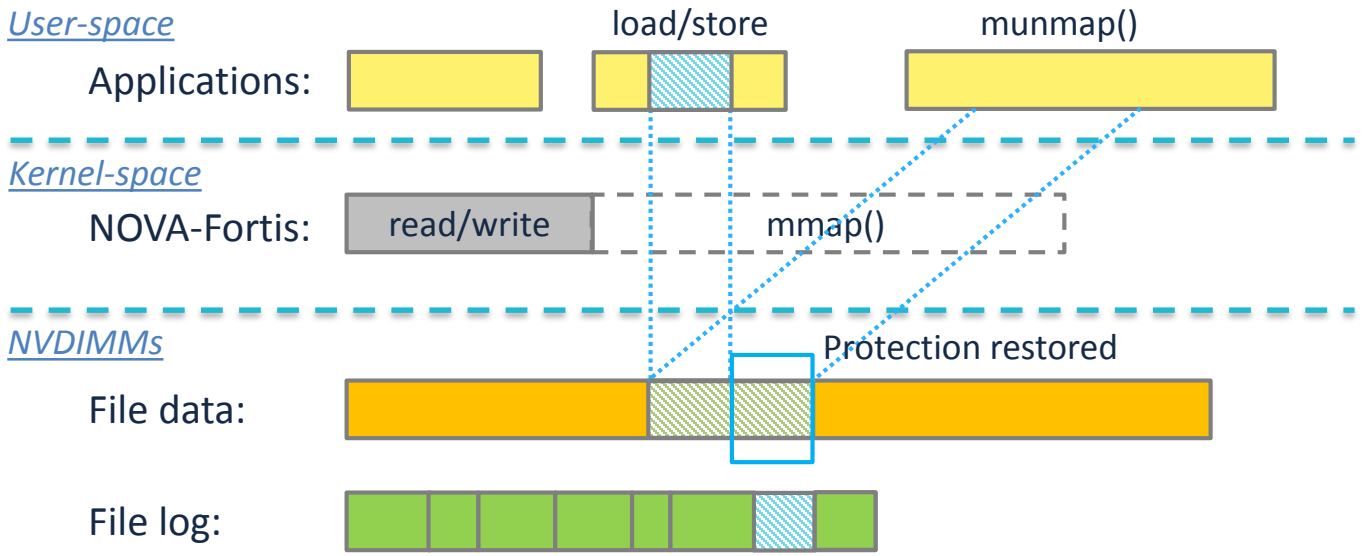
File data protection with DAX-mmap

- NOVA-Fortis logs mmap() operations



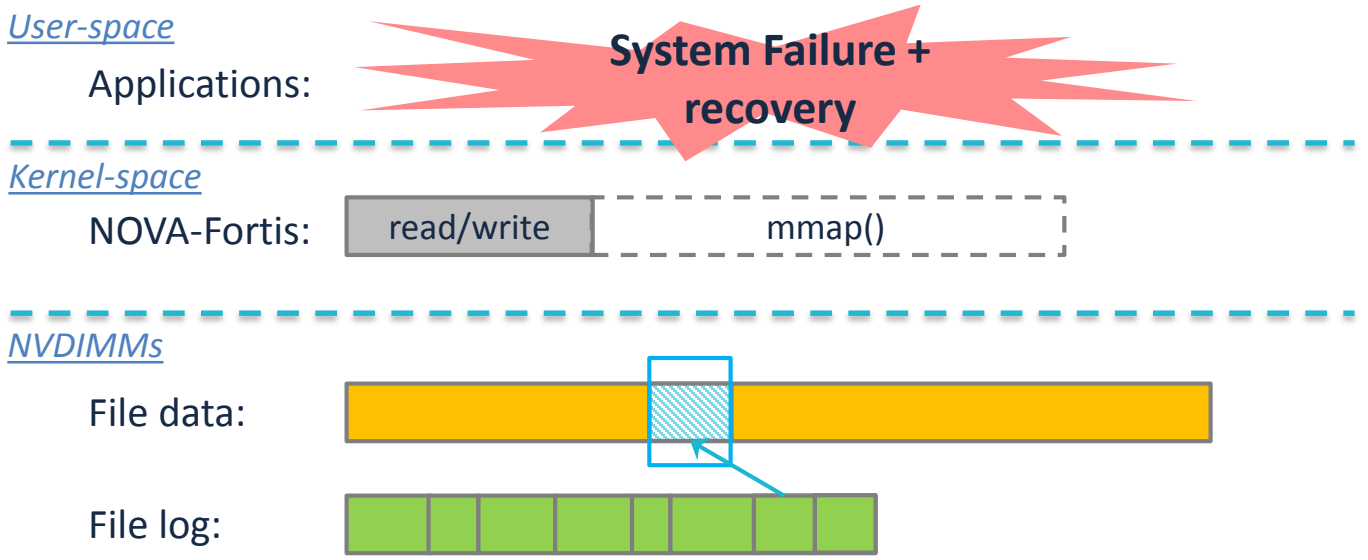
File data protection with DAX-mmap

- On munmap and during recovery, NOVA-Fortis restores protection



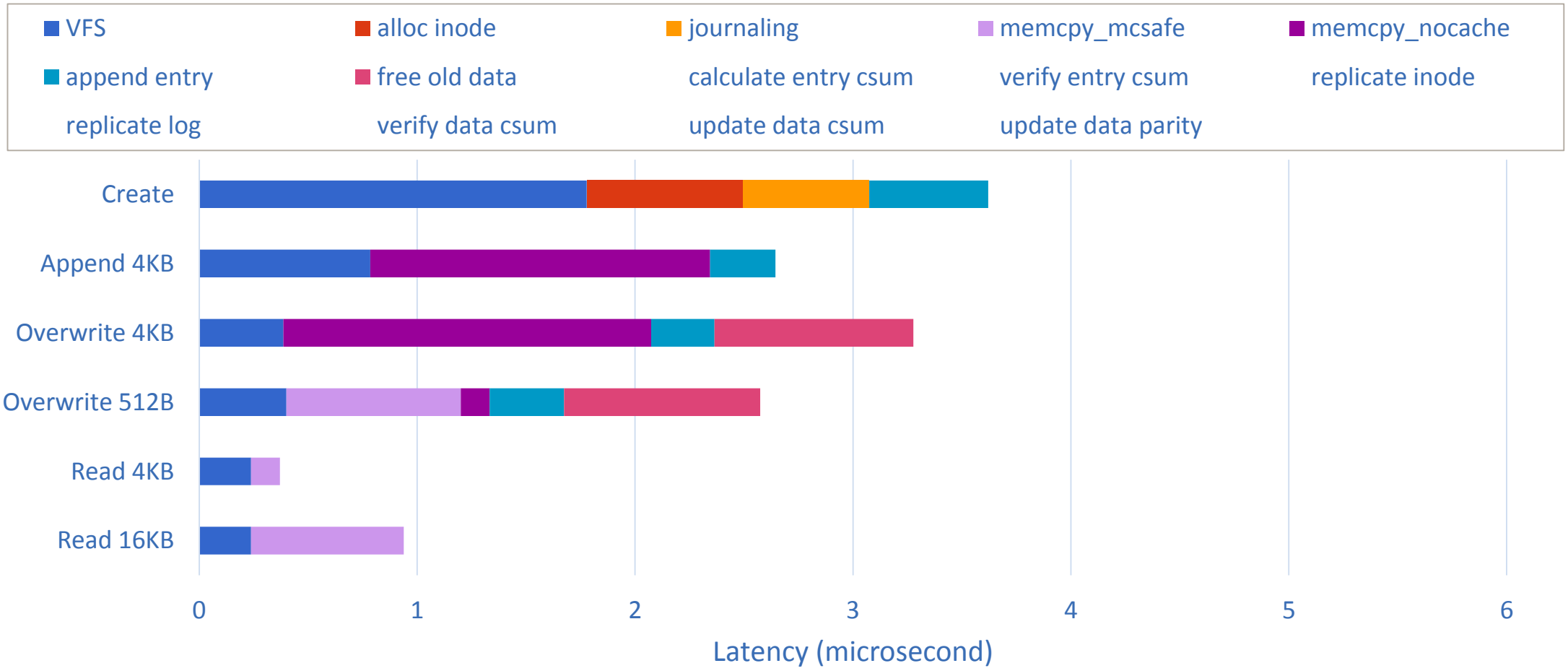
File data protection with DAX-mmap

- On munmap and during recovery, NOVA-Fortis restores protection

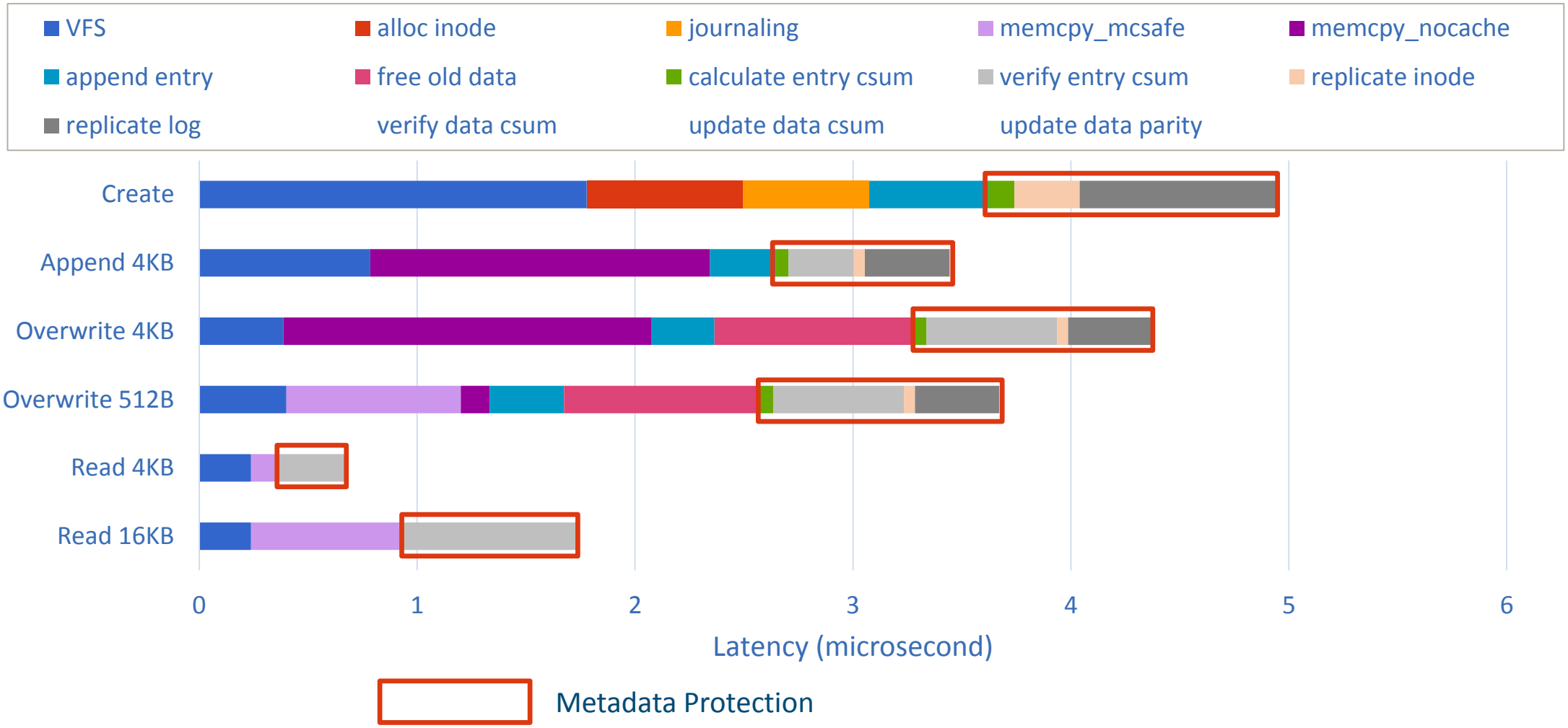


Performance

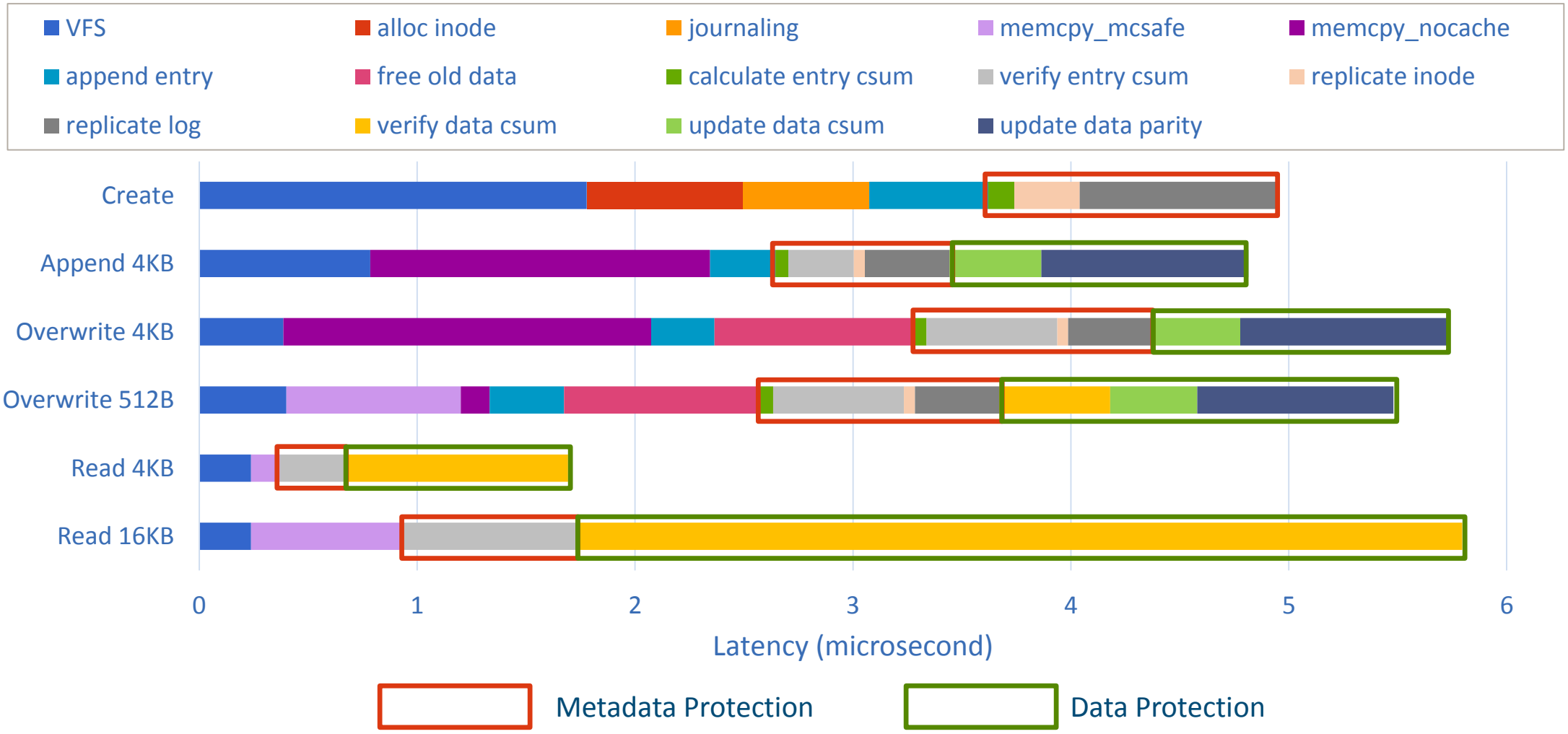
Latency breakdown



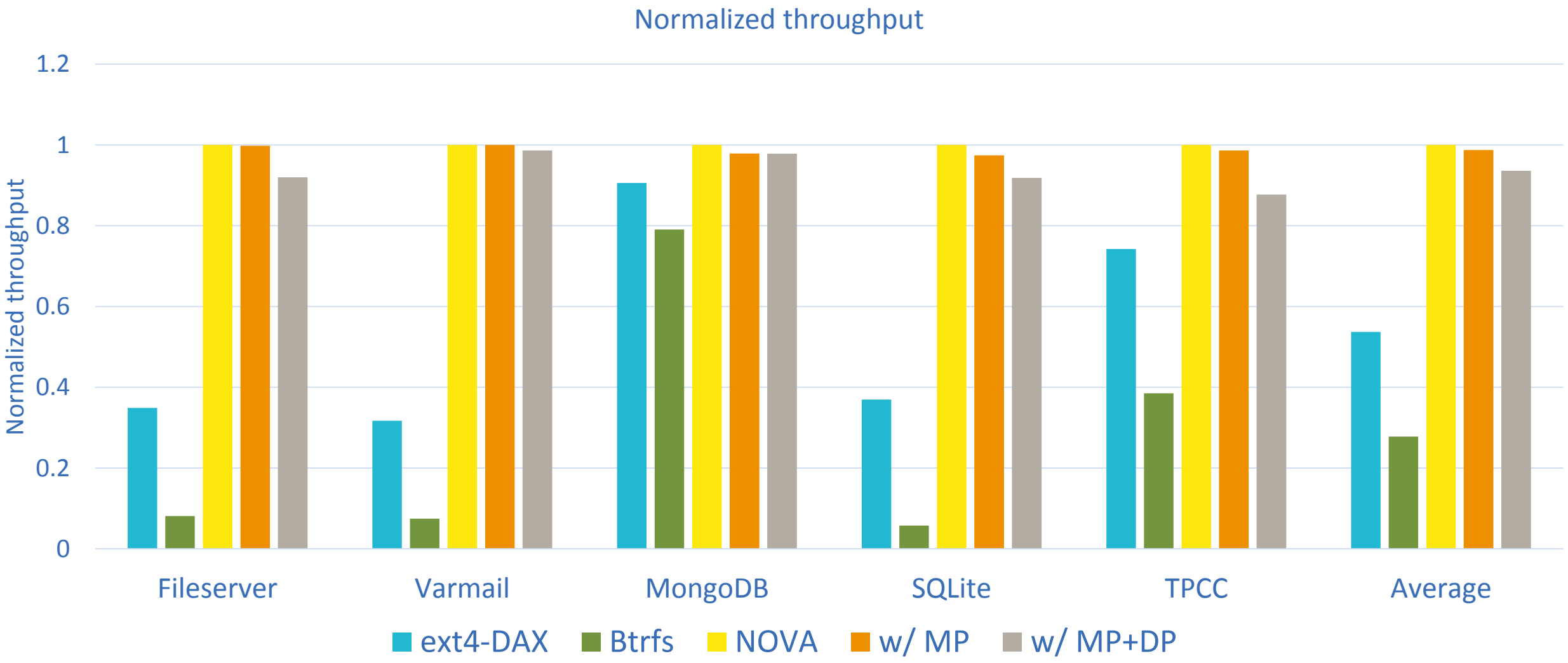
Latency breakdown



Latency breakdown



Application performance



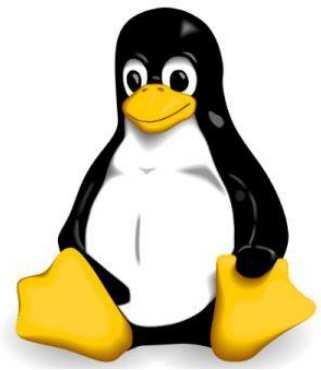
Conclusion

- Fault tolerance is critical for file system, but existing DAX file systems don’t provide it
- We identify new challenges that NVMM file system fault tolerance poses
- NOVA-Fortis provides fault tolerance with high performance
  - 1.5x on average to DAX-aware file systems without reliability features
  - 3x on average to other reliable file systems

NVSL

Give a try

https://github.com/NVSL/linux-nova



LWN.net
News from the source

Content
Weekly Edition
Archives

Andiry | Log out | [Subscribe to LWN]

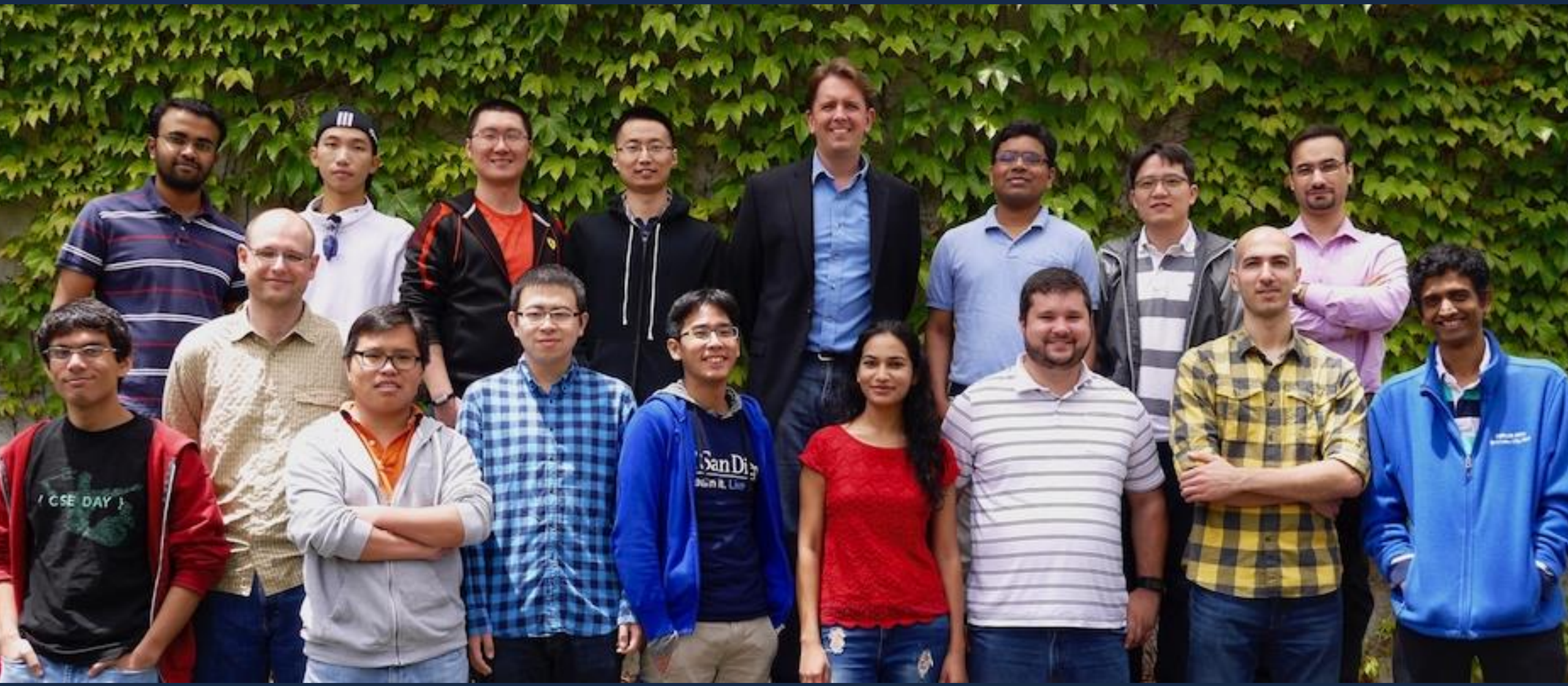
The NOVA filesystem

By Jonathan Corbet
August 4, 2017

Nonvolatile memory offers the promise of fast, byte-addressable storage that persists over power cycles. Taking advantage of that promise requires

NVSL

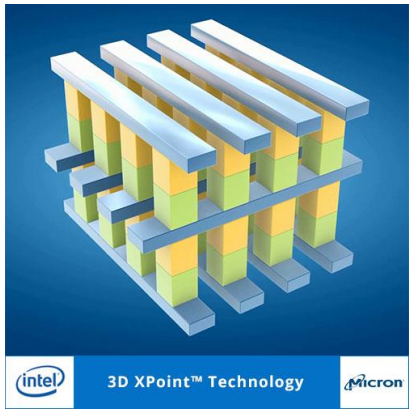
Thanks!

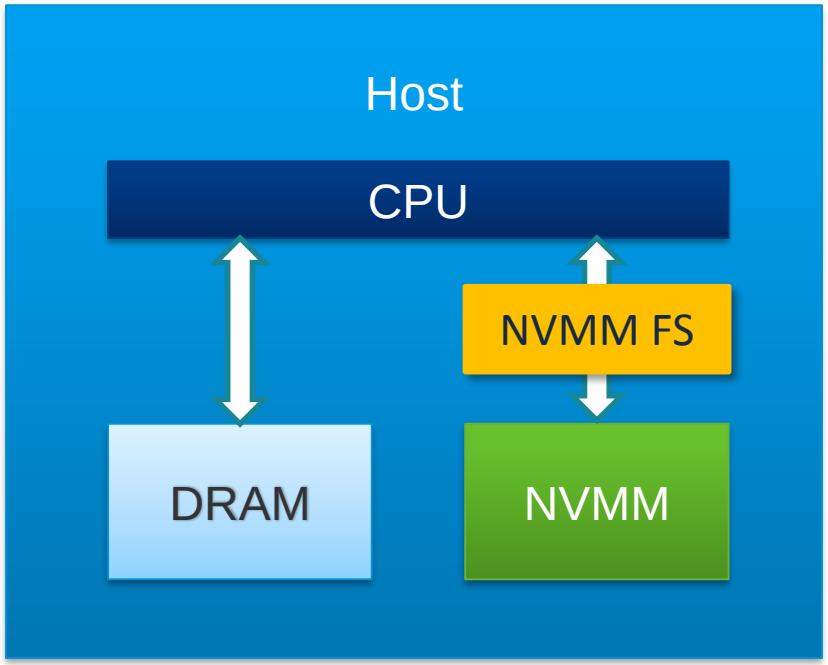


Backup slides

Hybrid DRAM/NVMM system

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





NVSL

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.

NVSL

Why LFS?

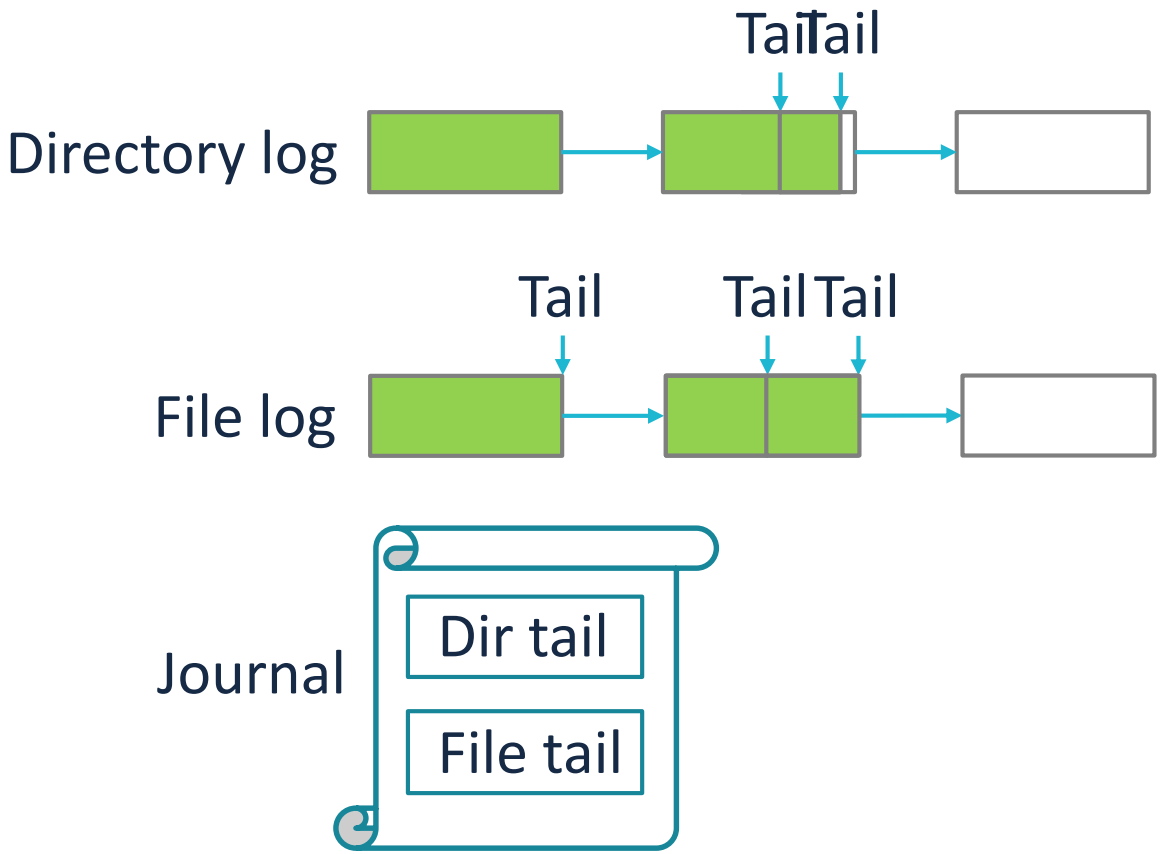
- Log-structuring provides cheaper atomicity than journaling and shadow paging
- NVMM supports fast, highly concurrent random accesses
  - Using multiple logs does not negatively impact performance
  - Log does not need to be contiguous

- Rethink and redesign log-structuring entirely

NVSL

Atomicity

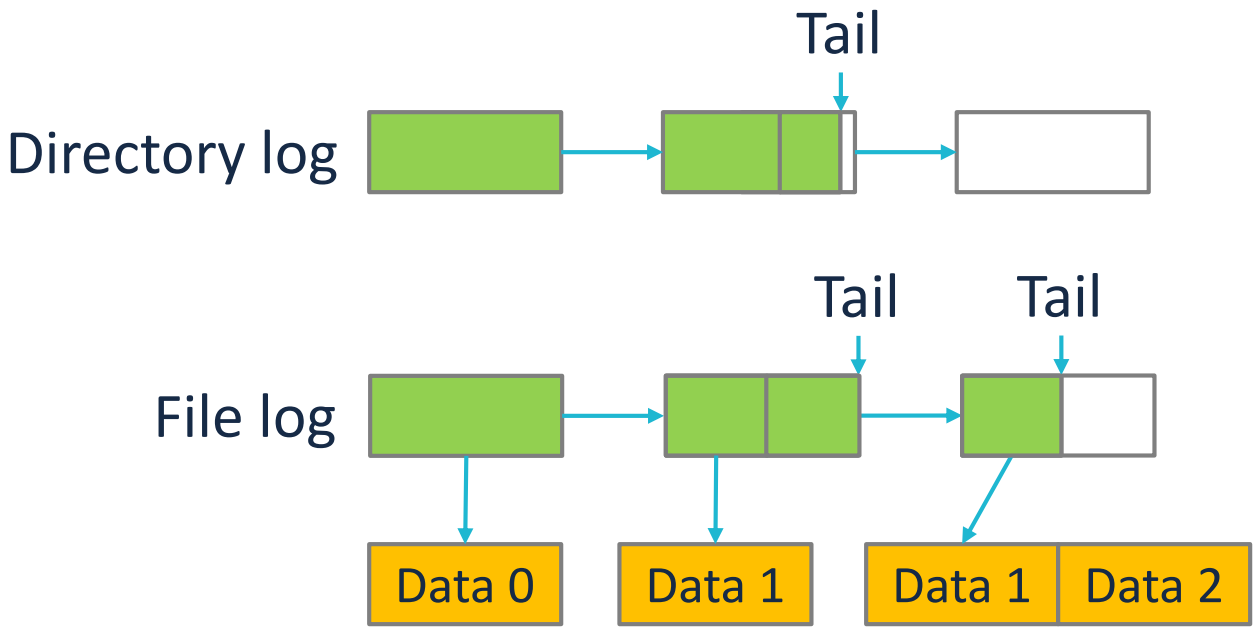
- Log-structuring for single log update
  - Write, msync, chmod, etc
  - Strictly commit log entry to NVMM before updating log tail
- Lightweight journaling for update across logs
  - Unlink, rename, etc
  - Journal log tails instead of metadata or data
- Copy-on-write for file data
  - Log only contains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



NVSL

Atomicity

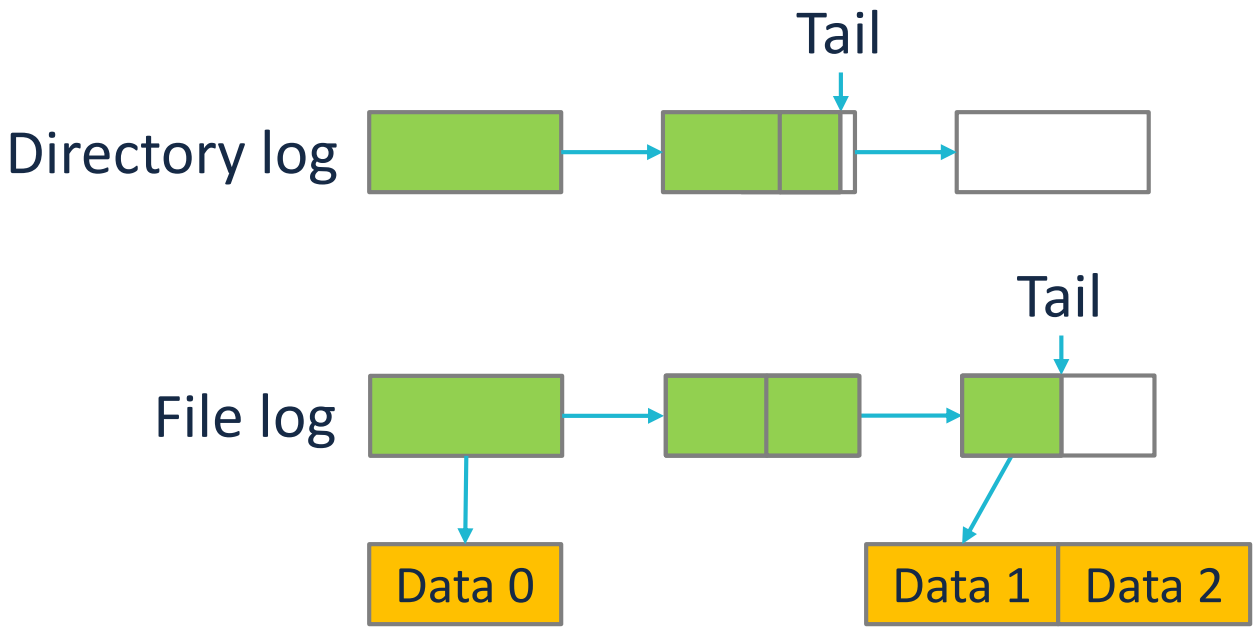
- Log-structuring for single log update
  - Write, msync, chmod, etc
  - Strictly commit log entry to NVMM before updating log tail
- Lightweight journaling for update across logs
  - Unlink, rename, etc
  - Journal log tails instead of metadata or data
- Copy-on-write for file data
  - Log only contains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



NVSL

Performance

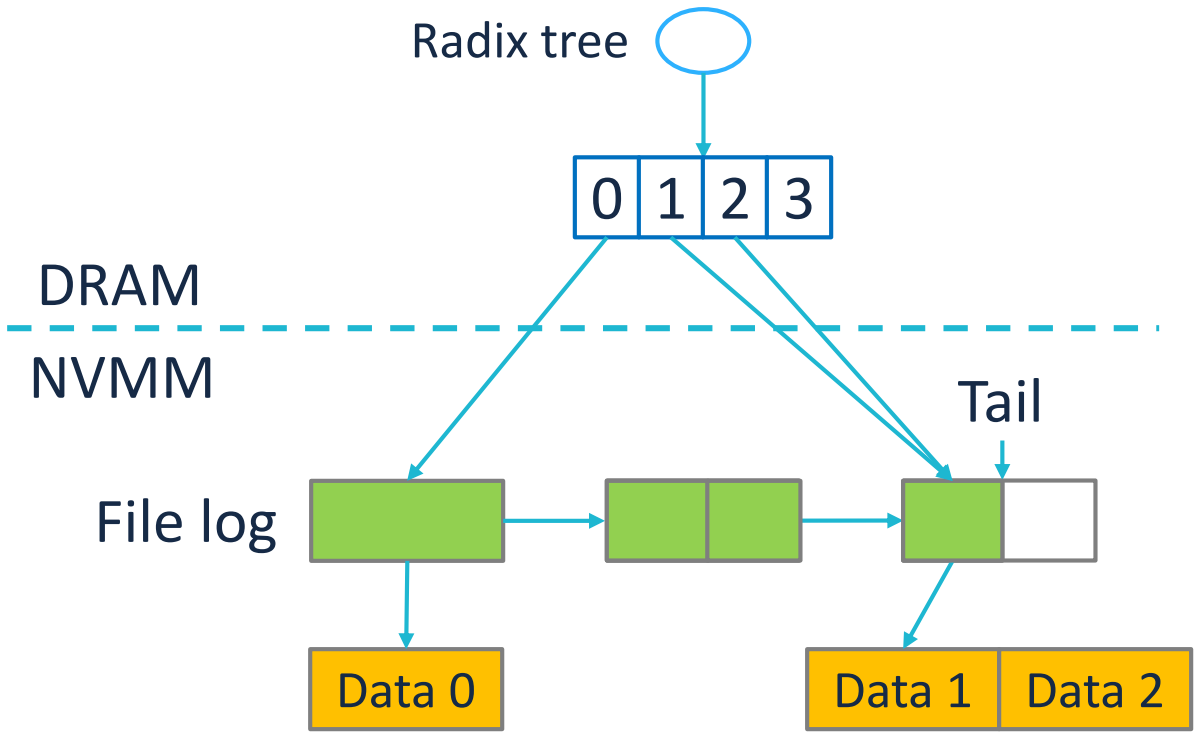
- Per-inode logging allows for high concurrency
- Split data structure between DRAM and NVMM
  - Persistent log is simple and efficient
  - Volatile tree structure has no consistency overhead



NVSL

Performance

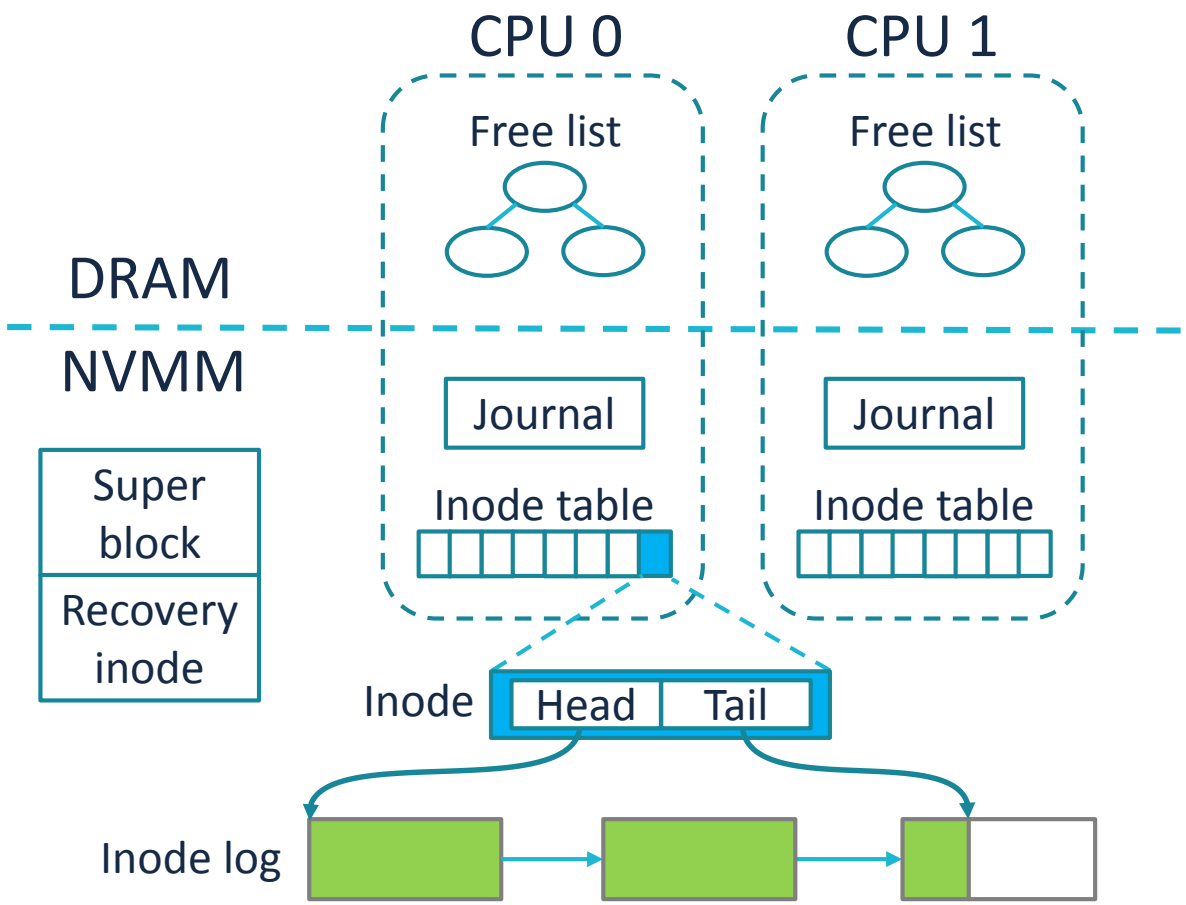
- Per-inode logging allows for high concurrency
- Split data structure between DRAM and NVMM
  - Persistent log is simple and efficient
  - Volatile tree structure has no consistency overhead



NVSL

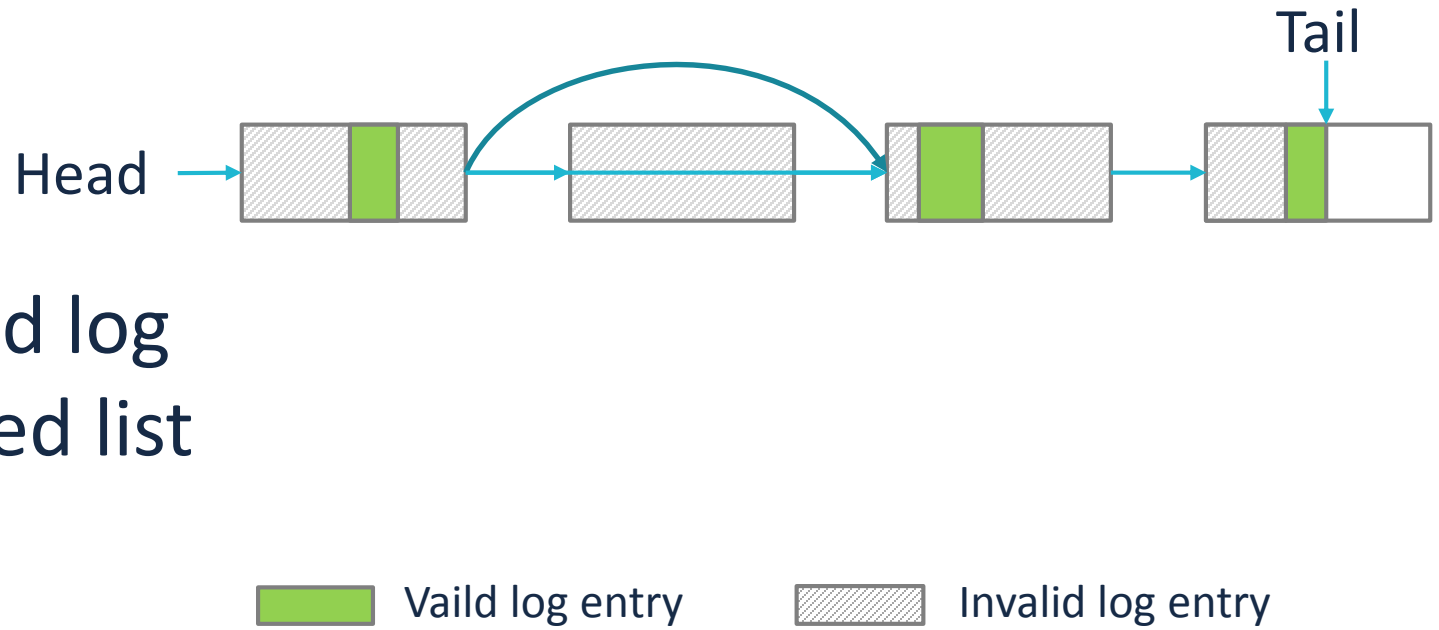
NOVA layout

- 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

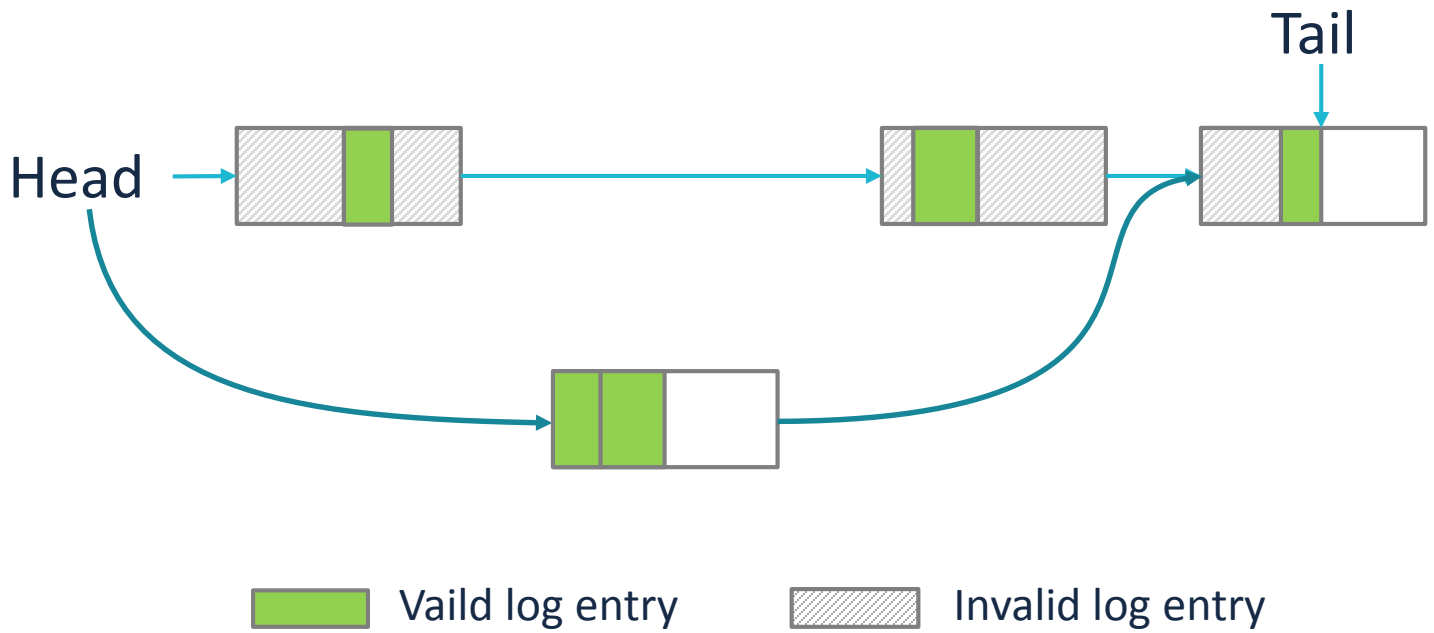


- Fast GC deletes dead log pages from the linked list
- No copying

NVSL

Thorough garbage collection

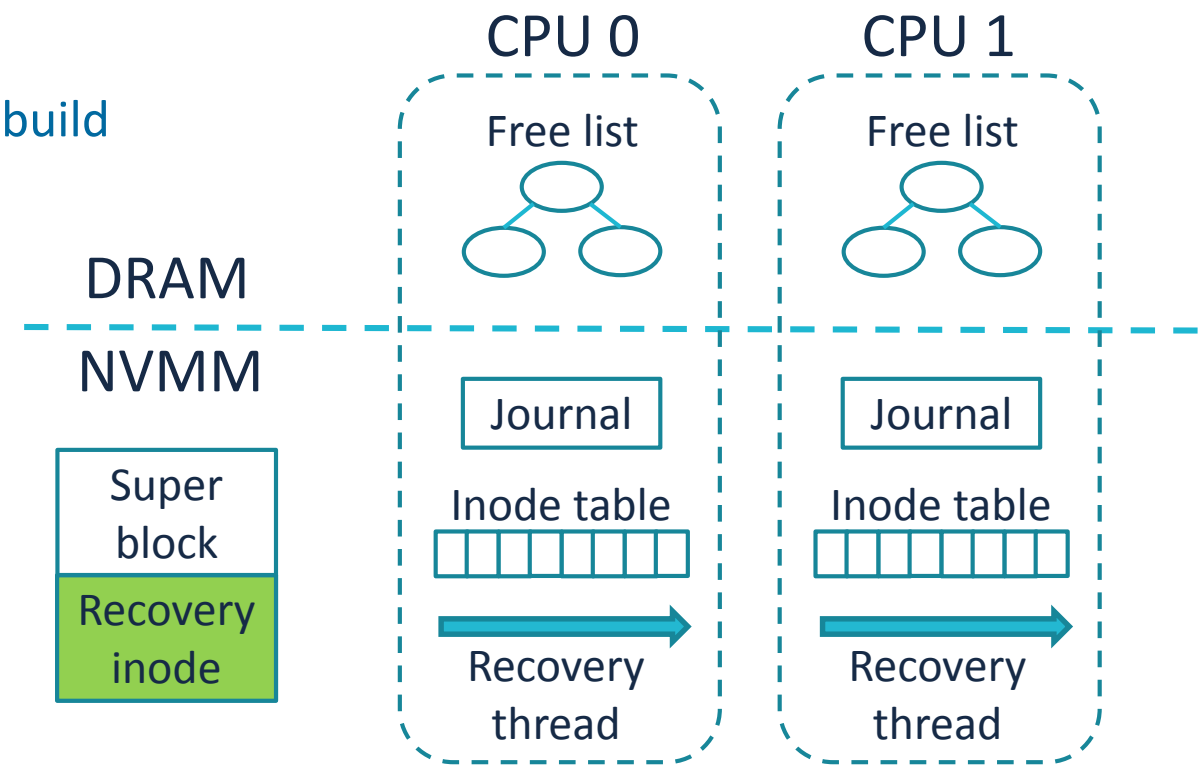
- Starts if valid log entries < 50% log length
- Format a new log and atomically replace the old one
- Only copy metadata



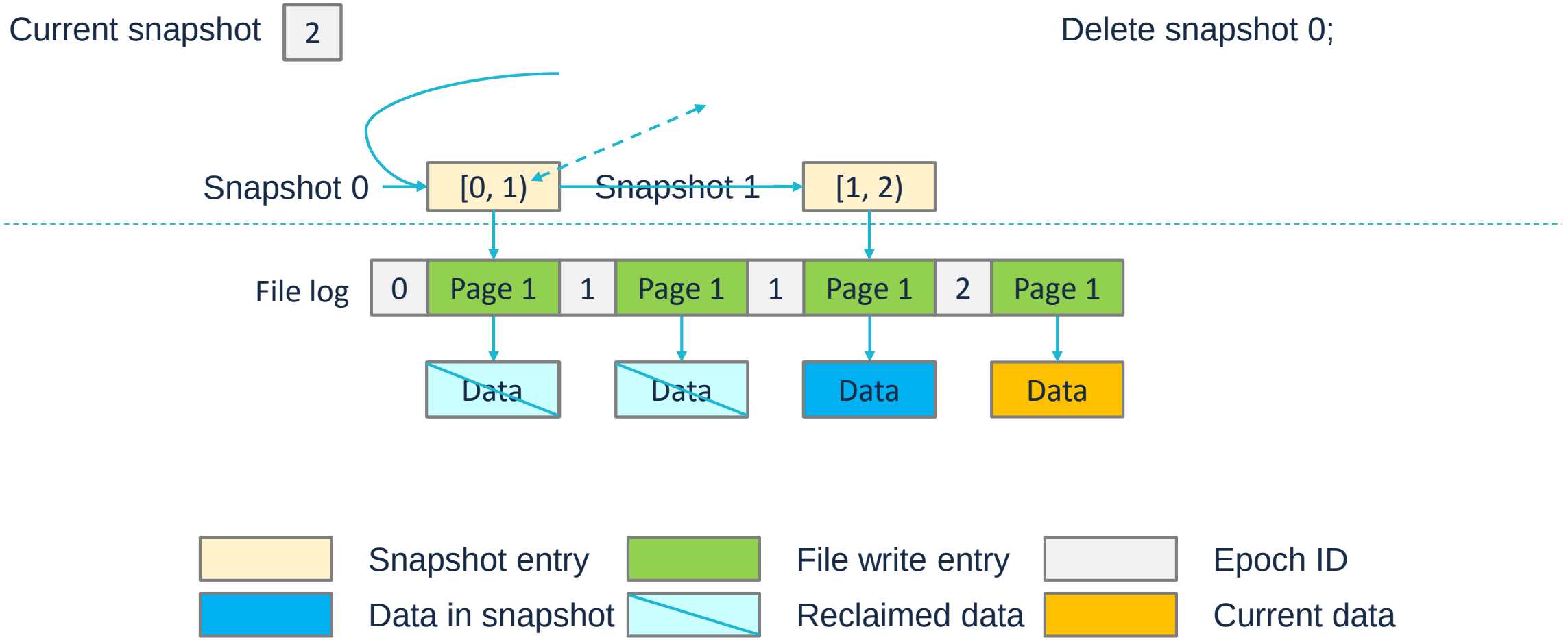
NVSL

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

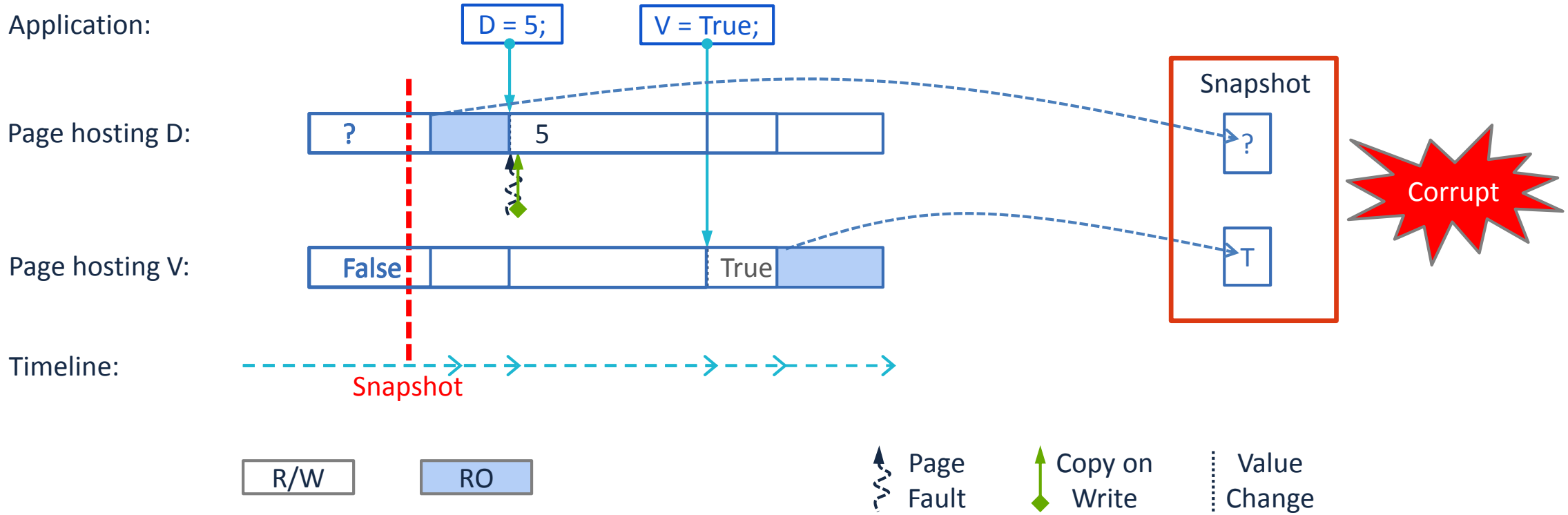


Snapshot for normal file I/O



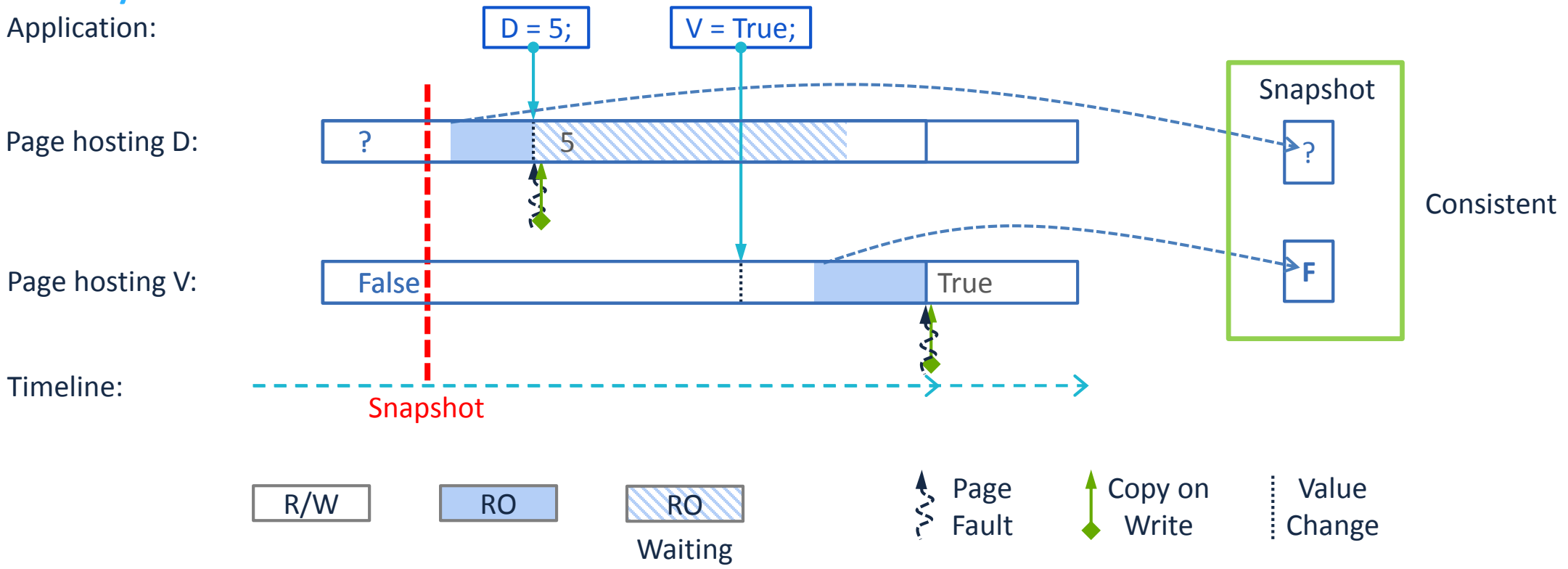
Corrupt Snapshots with DAX-mmap()

- Recovery invariant: if V == True, then D is valid
  - Incorrect: Naïvely mark pages read-only one-at-a-time

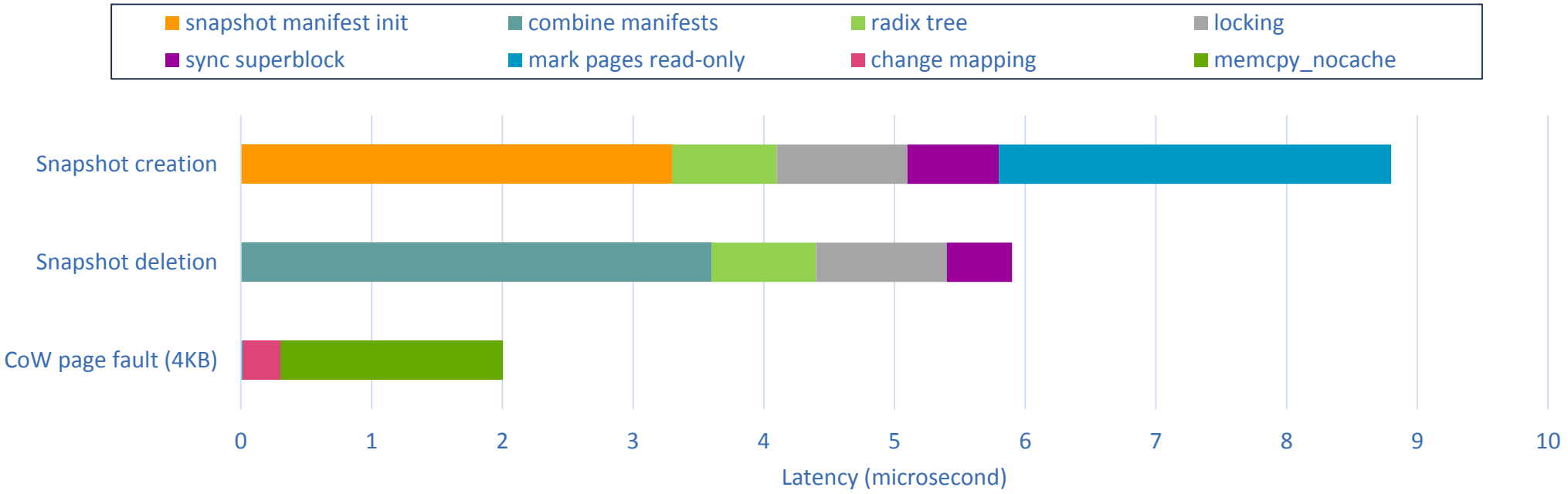


Consistent Snapshots with DAX-mmap()

- Recovery invariant: if V == True, then D is valid
  - Correct: Delay CoW page faults completion until all pages are read-only



Snapshot-related latency



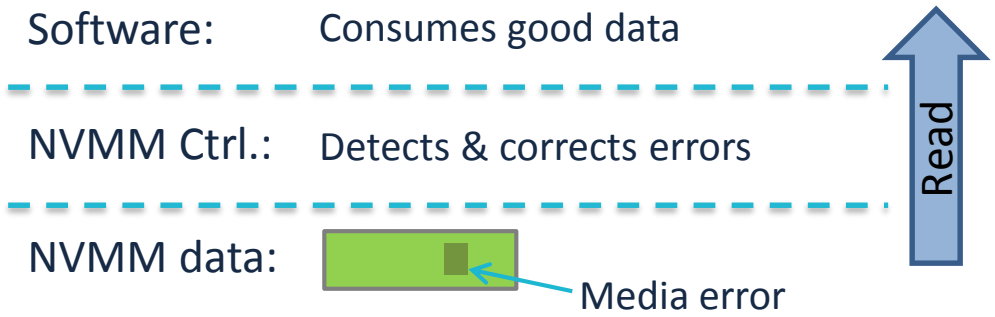
Defense Against Scribbles

- Tolerating Larger Scribbles
  - Allocate replicas far from one another
  - NOVA metadata can tolerate scribbles of 100s of MB
- Preventing scribbles
  - Mark all NVMM as read-only
  - Disable CPU write protection while accessing NVMM
  - Exposes all kernel data to bugs in a very small section of NOVA code.

NVSL

NVMM Failure Modes: Media Failures

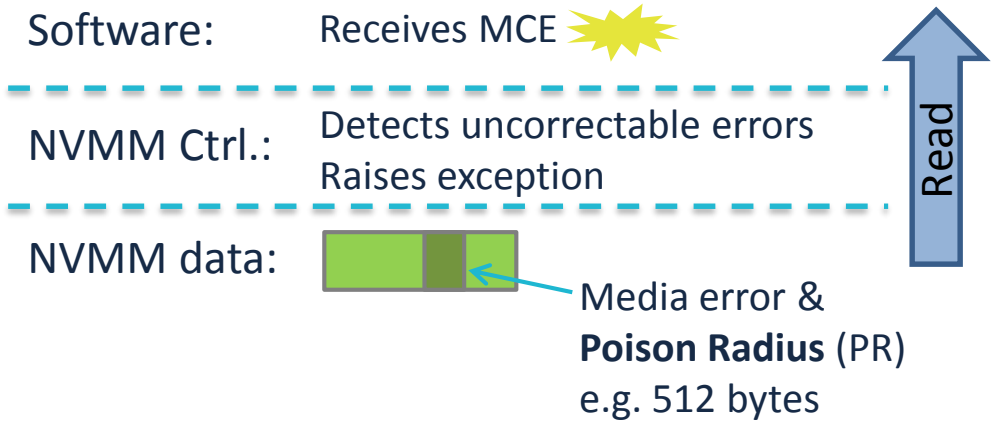
- Media errors
  - Detectable & correctable
  - Detectable & uncorrectable
  - Undetectable
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware



NVSL

NVMM Failure Modes: Media Failures

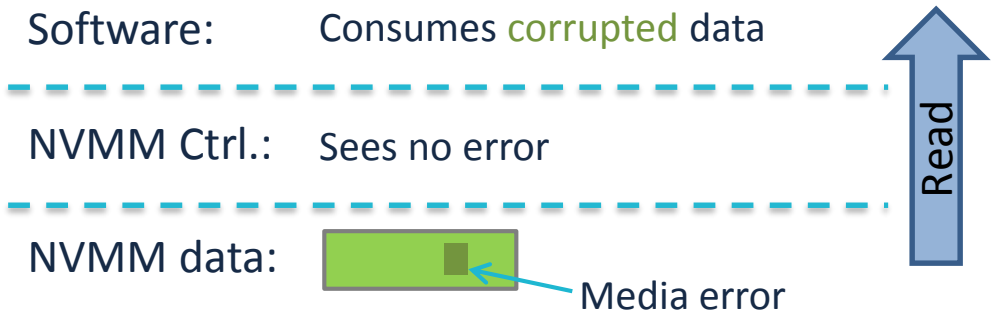
- Media errors
  - Detectable & correctable
  - Detectable & uncorrectable
  - Undetectable
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware



NVSL

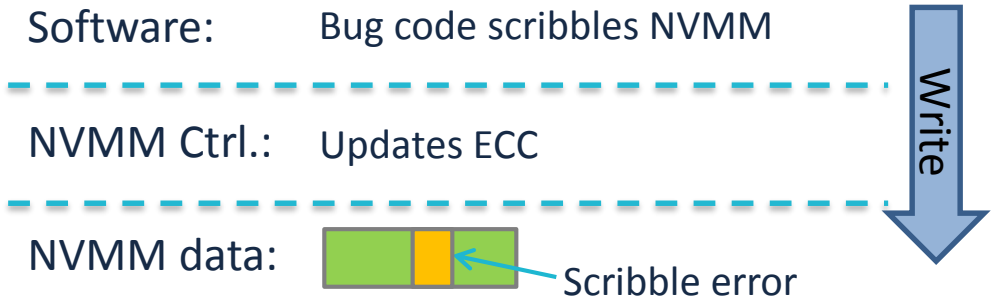
NVMM Failure Modes: Media Failures

- Media errors
  - Detectable & correctable
  - Detectable & uncorrectable
  - Undetectable
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware



NVMM Failure Modes: Scribbles

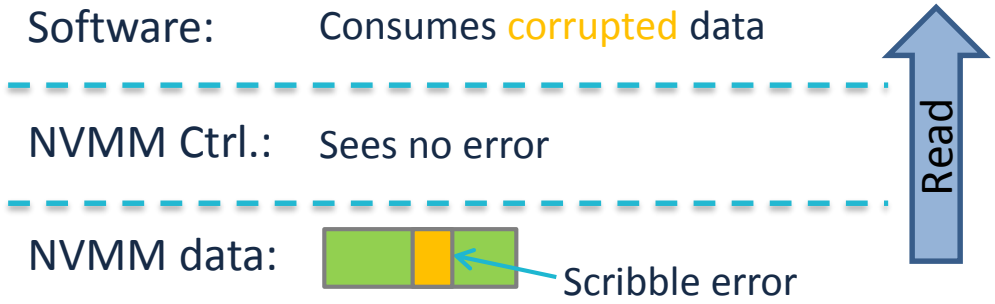
- Media errors
  - Detectable & correctable
  - Detectable & uncorrectable
  - Undetectable
- Software “scribbles”
  - Kernel bugs or NOVA bugs
  - NVMM file systems are highly vulnerable



NVSL

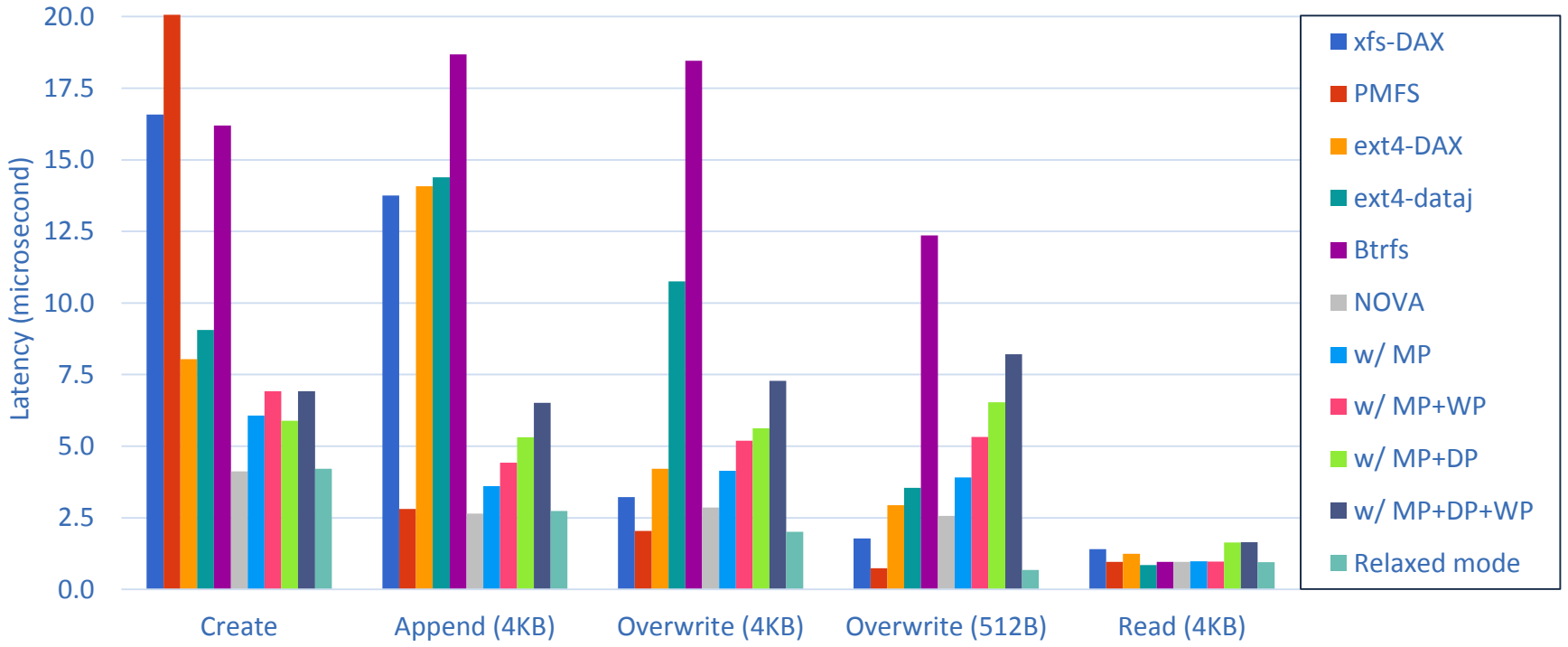
NVMM Failure Modes: Scribbles

- Media errors
  - Detectable & correctable
  - Detectable & uncorrectable
  - Undetectable
- Software "scribbles"
  - Kernel bugs or NOVA bugs
  - NVMM file systems are highly vulnerable

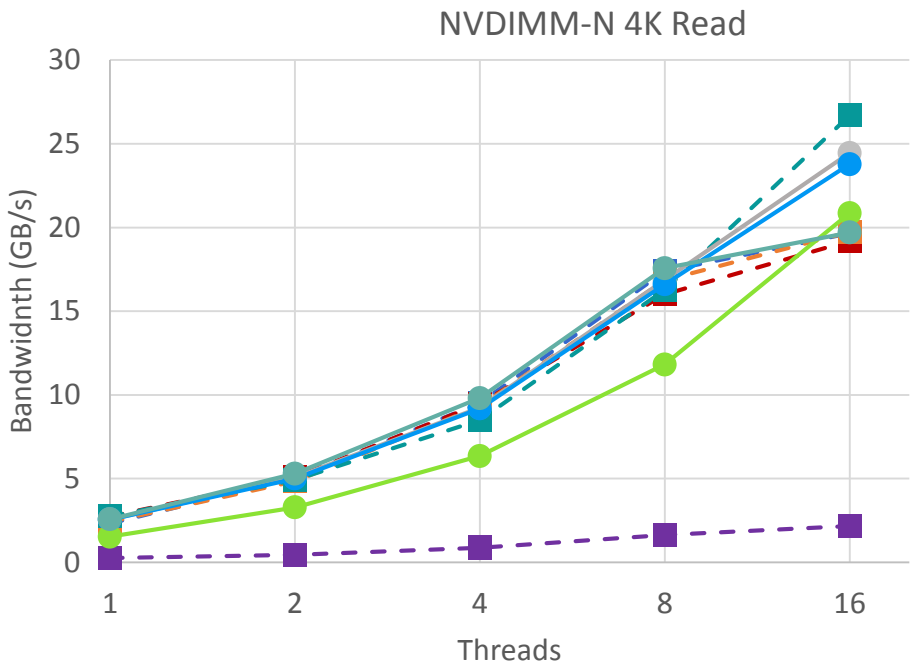


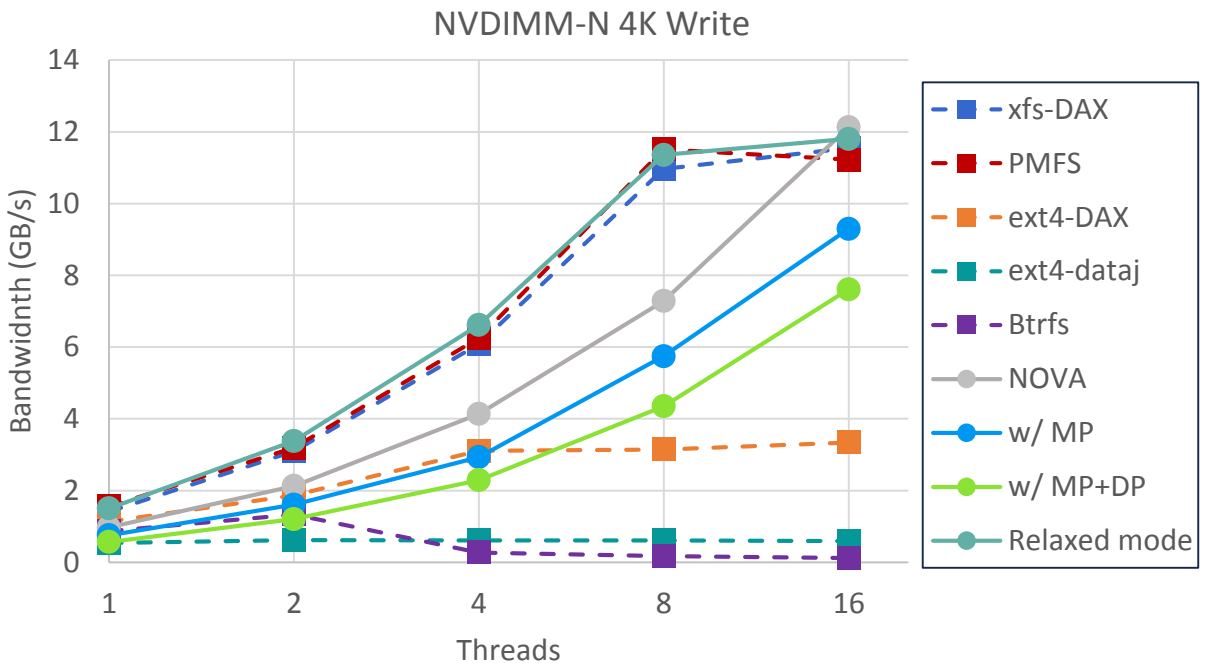
NVSL

File operation latency

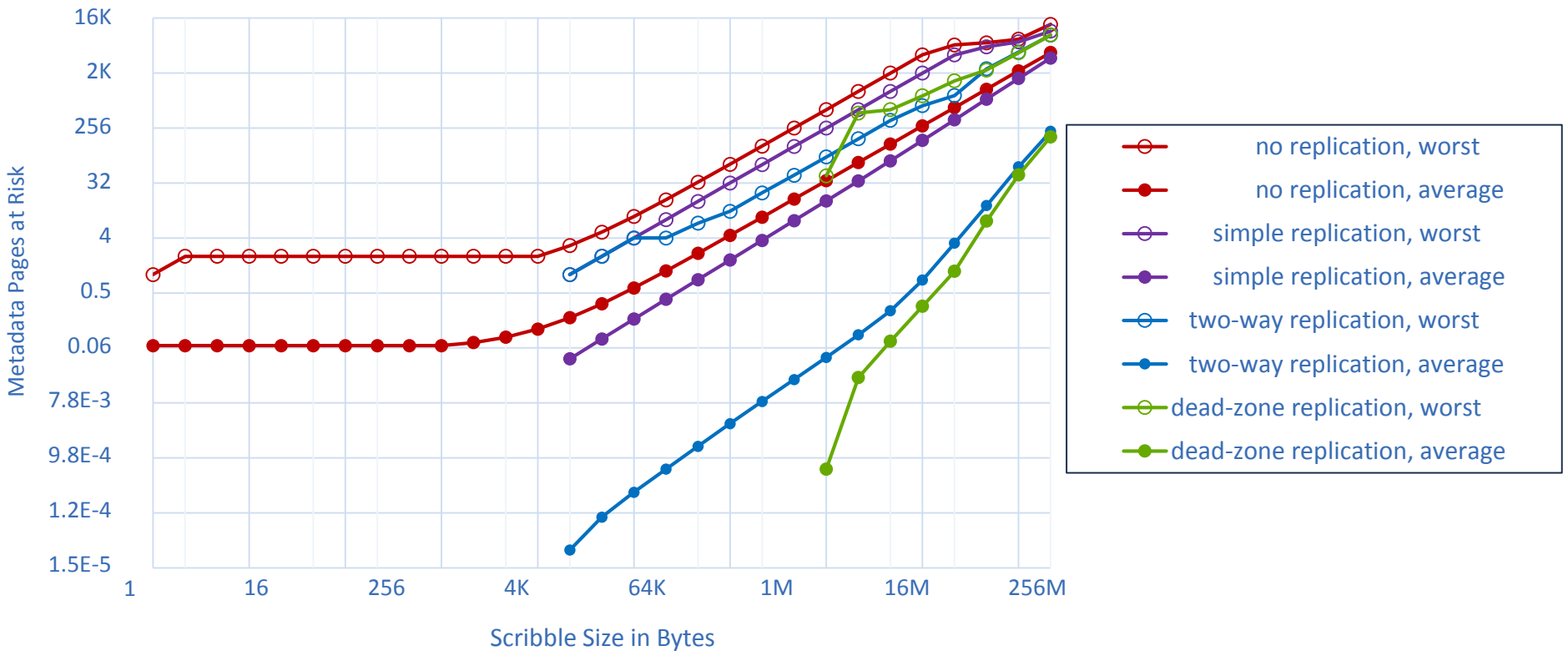


Random R/W bandwidth on NVDIMM-N

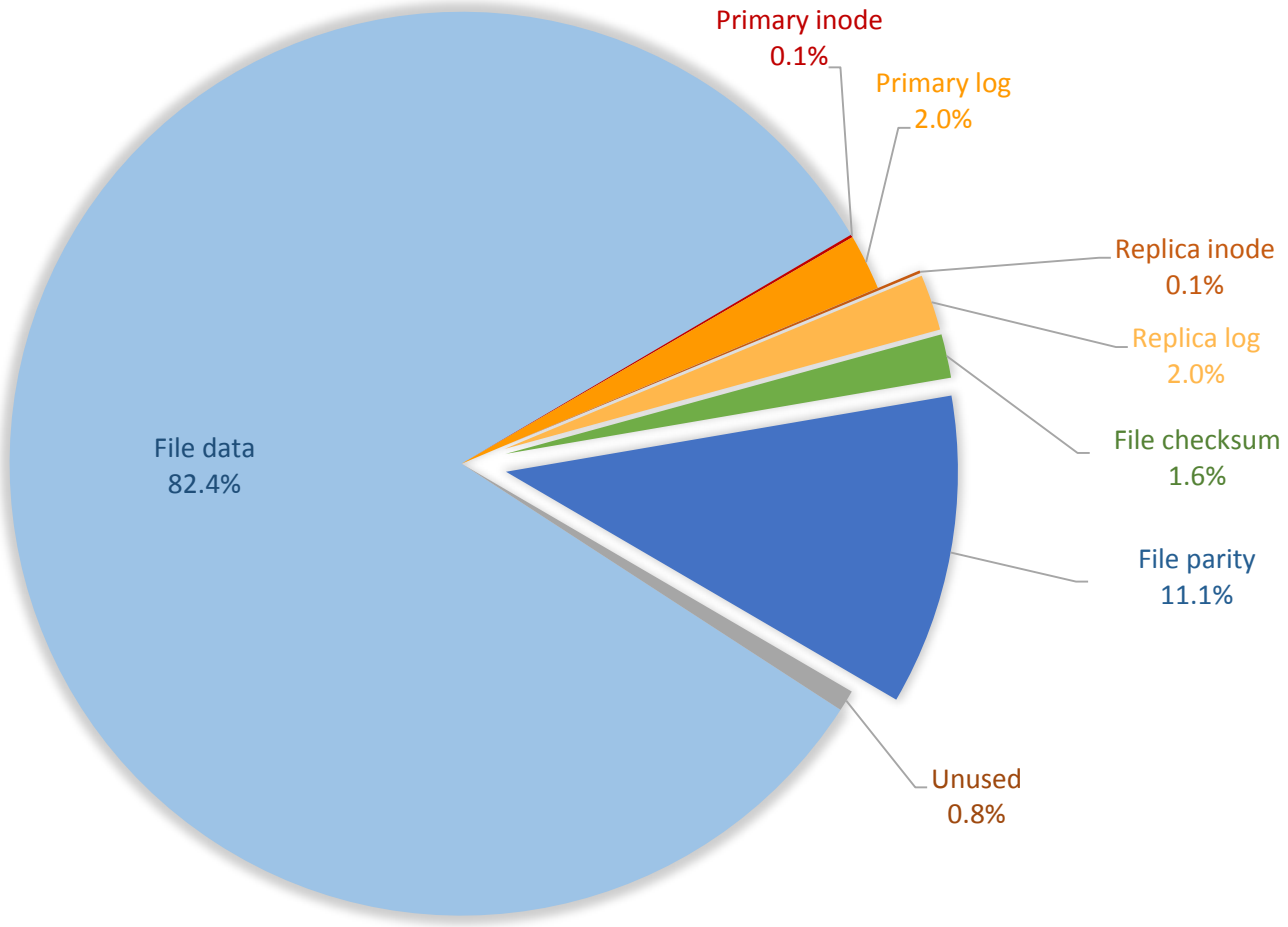




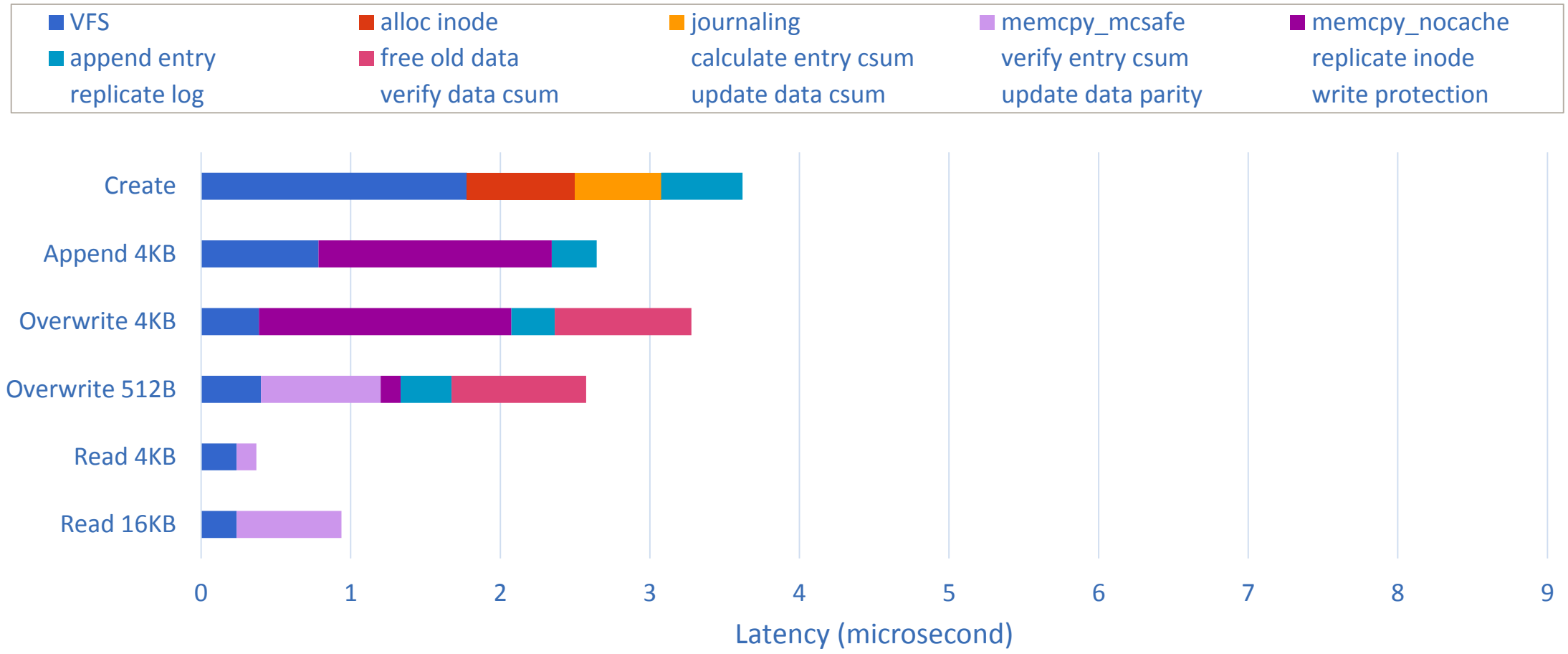
Scribble size and metadata bytes at risk



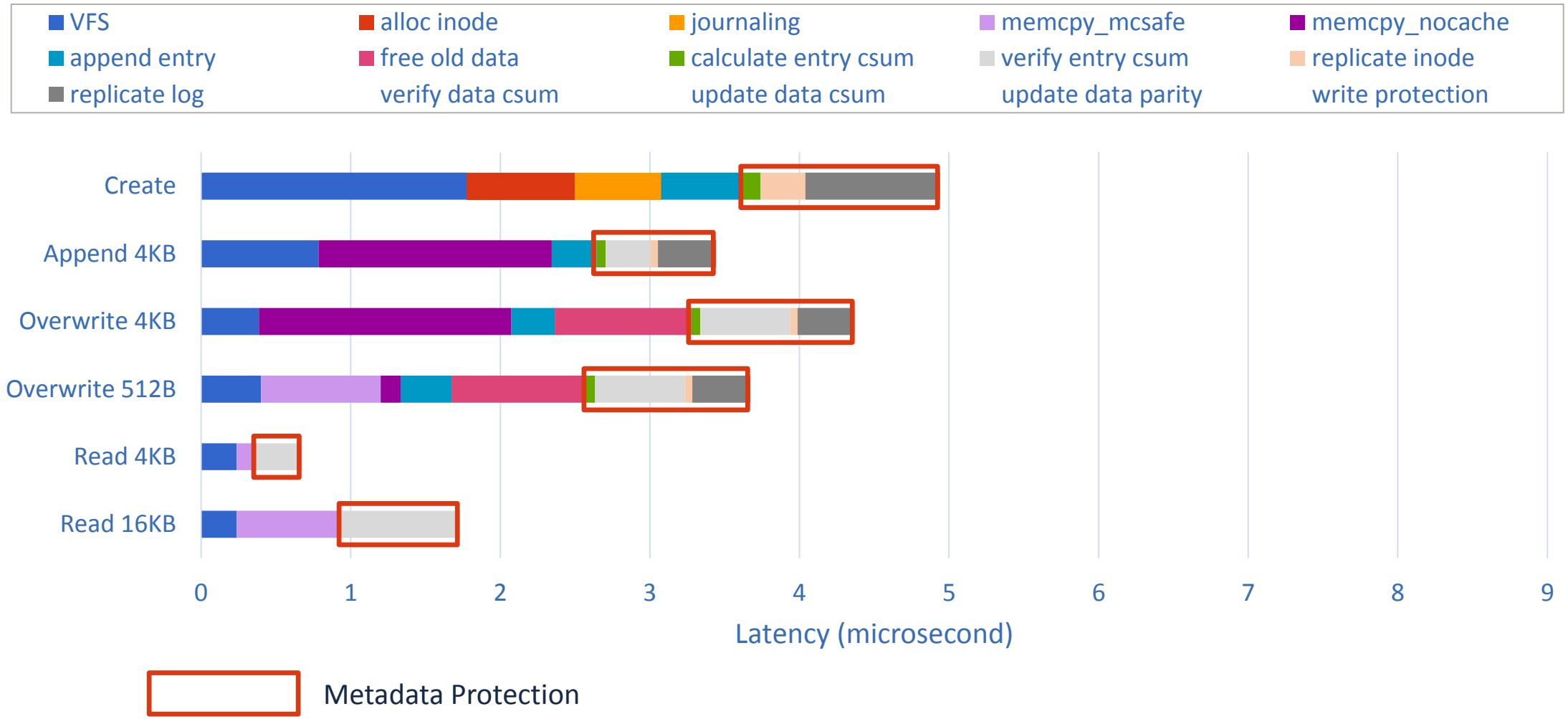
Storage overhead



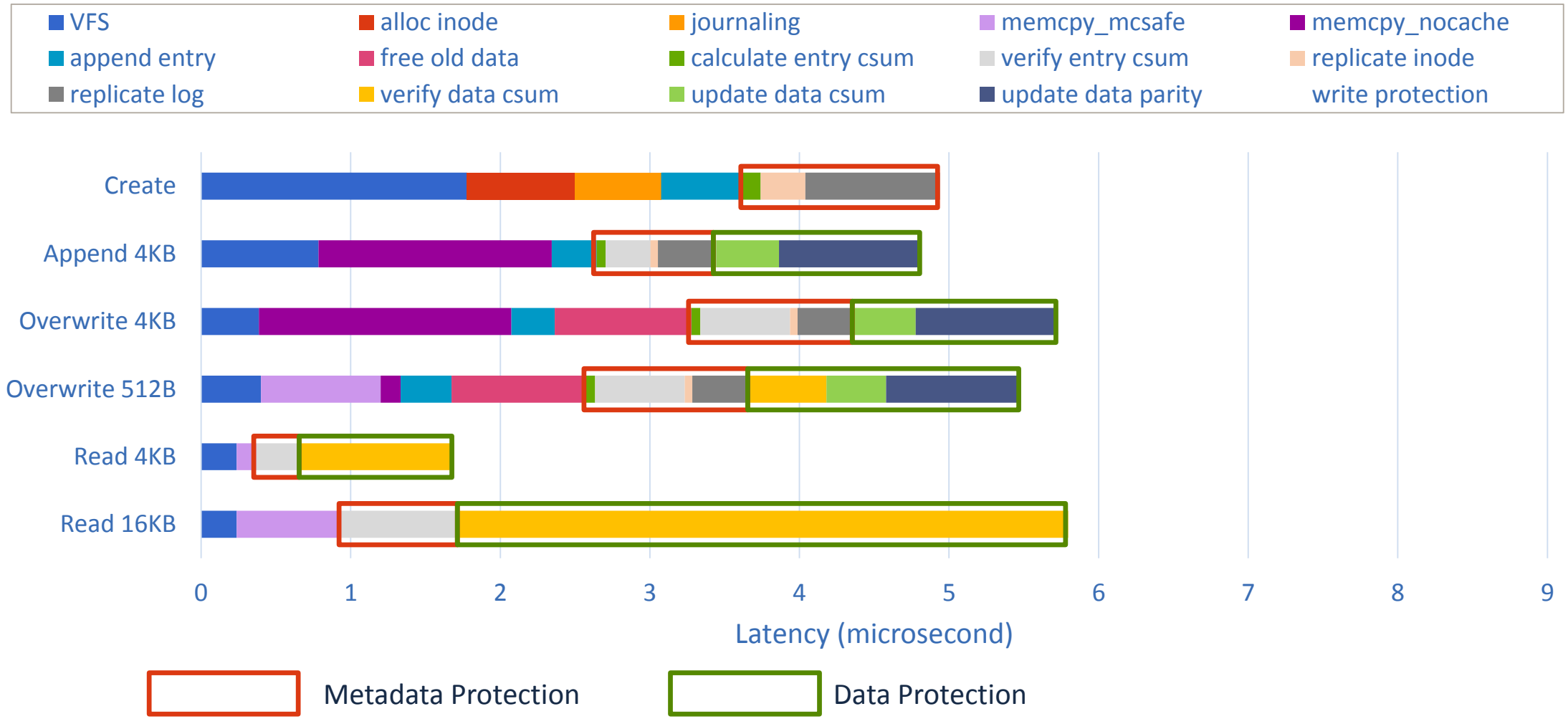
Latency breakdown



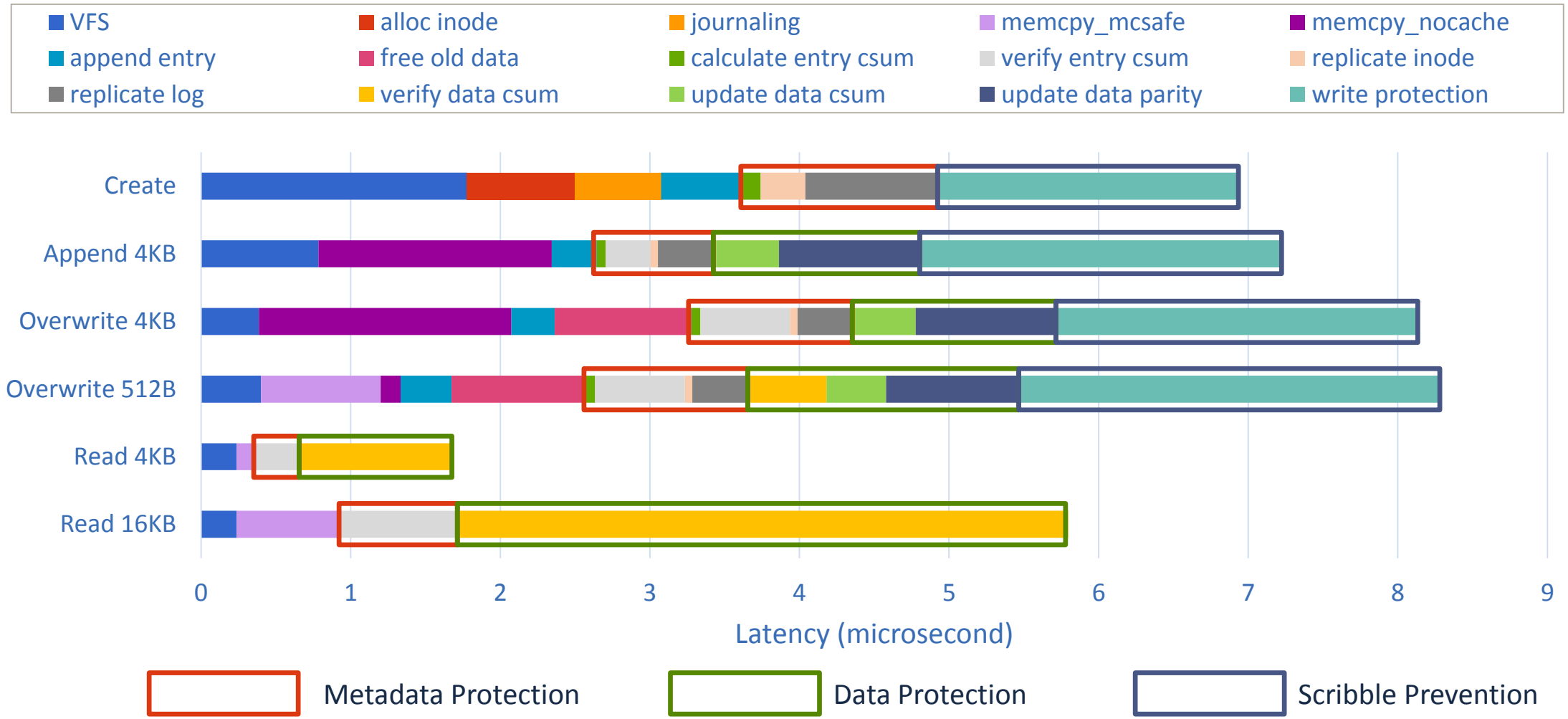
Latency breakdown



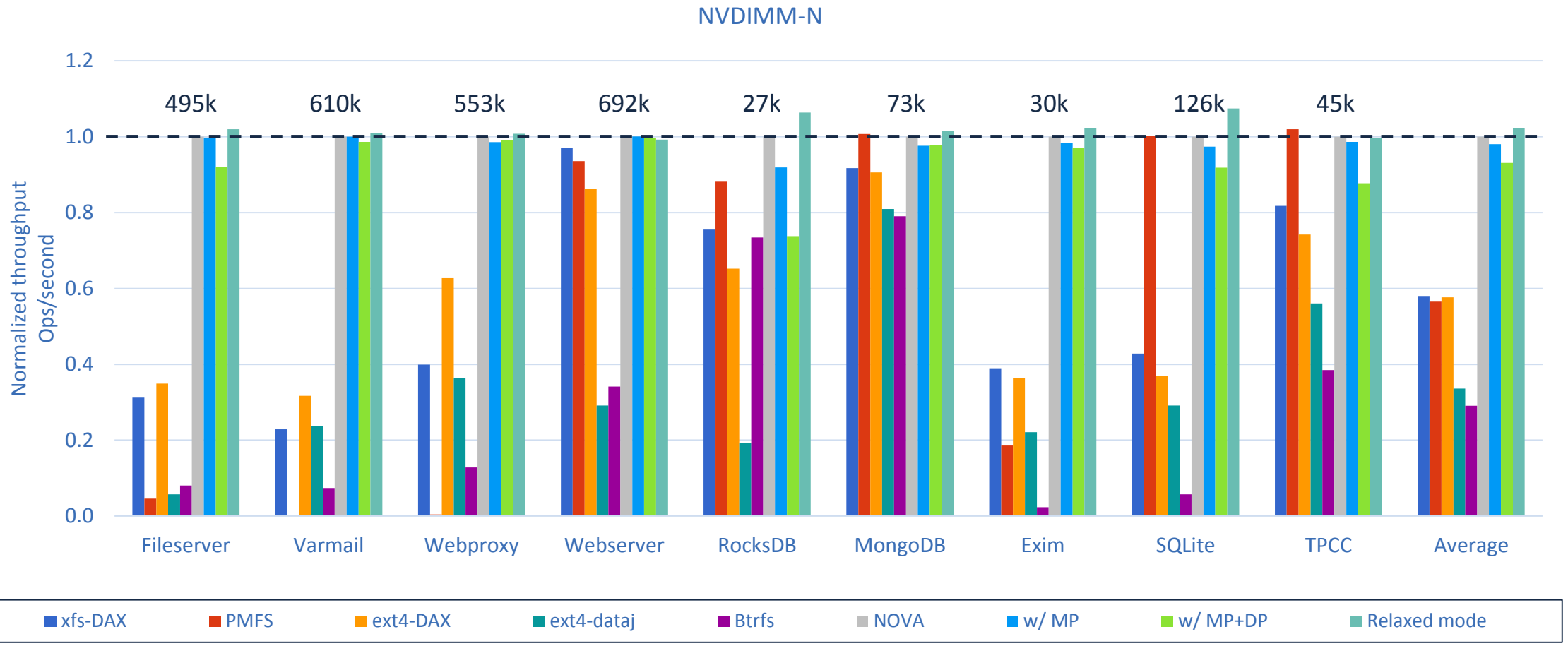
Latency breakdown



Latency breakdown



Application performance on NOVA-Fortis



Application performance on NOVA-Fortis