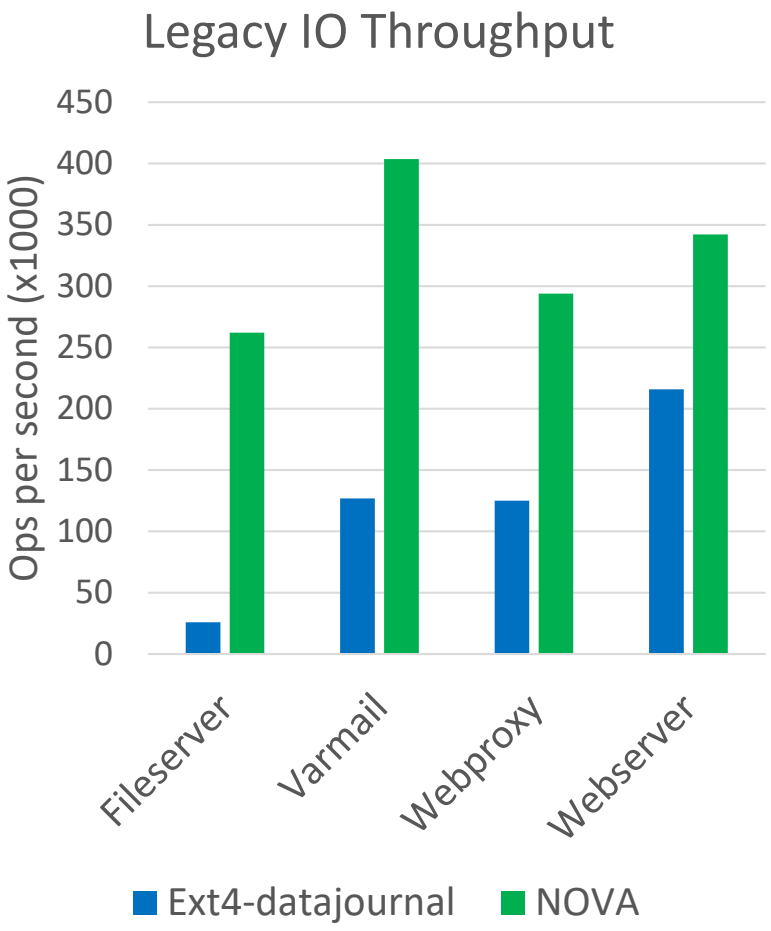
NOVA: A High-Performance, Hardened File System for Non-Volatile Main Memories

Jian 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

NVDIMM Usage Models

- Legacy File IO Acceleration – fast and easy
  - Run existing IO-intensive apps on NVDIMMs
  - "just works"
  - *NOVA is 30% - 10x faster than Ext4 for write intensive workloads.*
  - *Need strong protections on data.*
- DAX Mmap -- maximum speed + programming challenges
  - Load-store access
  - *You still need a strongly-consistent file system*
    - File system corruption can still destroy your data
    - NOVA is strongly consistent
  - *Data protection is still critical*



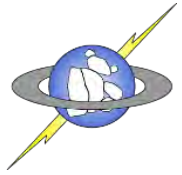
XFS F2FS NILFS

EXT4 BTRFS

Disk-based file systems are inadequate for NVMM

- Disk-based file systems cannot exploit NVMM performance
- Performance optimization compromises consistency on system failure [1]

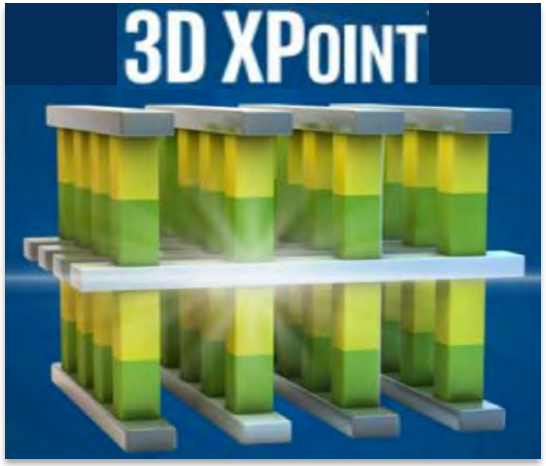
	Atomicity						Data Protection		
	1-Sector overwrite	1-Sector append	1-Block overwrite	1-Block append	N-Block overwrite	N-Block append	Data	Meta-data	Snap-shots
Ext4 wb	✓	X	X	X	X	X	X	✓	✓
Ext4 Order	✓	✓	X	✓	X	✓	X	✓	✓
Ext4 Dataj	✓	✓	✓	✓	X	✓	X	✓	✓
Btrfs	✓	✓	✓	✓	X	✓	✓	✓	✓
xfs	✓	✓	X	✓	X	✓	X	✓	✓
Reiserfs	✓	✓	X	✓	X	✓	X	✓	✓



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

BPFS SCMFS PMFS Aerie

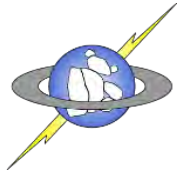
EXT4-DAX M1FS XFS-DAX



NVMM file systems don't provide strong consistency or data protection

- DAX does not provide data atomicity guarantees
- Programming is more difficult

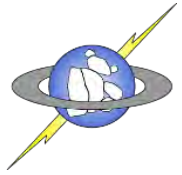
	Atomicity		Data Protection		
	Metadata	Data	Data	Meta-data	Snap-shots
BPFS	✓	✓	X	X	X
PMFS	✓	X	X	X	X
Ext4 DAX	✓	X	X	✓	X
XFS DAX	✓	X	X	✓	X
SCMFS	X	X	X	X	X
Aerie	✓	X	X	X	X



NOVA provides strong atomicity guarantee

	Atomicity						Data Protection		
	1-Sector overwrite	1-Sector append	1-Block overwrite	1-Block append	N-Block overwrite	N-Block append	Data	Meta-data	Snap-shots
Ext4 wb	✓	X	X	X	X	X	X	✓	✓
Ext4 Order	✓	✓	X	✓	X	✓	X	✓	✓
Ext4 Dataj	✓	✓	✓	✓	X	✓	X	✓	✓
Btrfs	✓	✓	✓	✓	X	✓	✓	✓	✓
xfs	✓	✓	X	✓	X	✓	X	✓	✓
Reiserfs	✓	✓	X	✓	X	✓	X	✓	✓

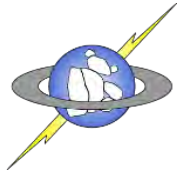
	Atomicity		Data Protection		
	Metadata	Data	Data	Meta-data	Snap-shots
BPFS	✓	✓	X	X	X
PMFS	✓	X	X	X	X
Ext4 DAX	✓	X	X	✓	X
XFS DAX	✓	X	X	✓	X
SCMFS	X	X	X	X	X
Aerie	✓	X	X	X	X



NOVA provides strong atomicity guarantee

	Atomicity						Data Protection		
	1-Sector overwrite	1-Sector append	1-Block overwrite	1-Block append	N-Block overwrite	N-Block append	Data	Meta-data	Snap-shots
Ext4 wb	✓	X	X	X	X	X	X	✓	✓
Ext4 Order	✓	✓	X	✓	X	✓	X	✓	✓
Ext4 Dataj	✓	✓	✓	✓	X	✓	X	✓	✓
Btrfs	✓	✓	✓	✓	X	✓	✓	✓	✓
xfs	✓	✓	X	✓	X	✓	X	✓	✓
Reiserfs	✓	✓	X	✓	X	✓	X	✓	✓
NOVA	✓	✓	✓	✓	✓	✓	✓	✓	✓

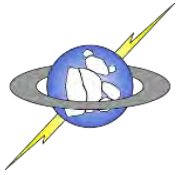
	Atomicity		Data Protection		
	Metadata	Data	Data	Meta-data	Snap-shots
BPFS	✓	✓	X	X	X
PMFS	✓	X	X	X	X
Ext4 DAX	✓	X	X	✓	X
XFS DAX	✓	X	X	✓	X
SCMFS	X	X	X	X	X
Aerie	✓	X	X	X	X
NOVA	✓	✓	✓	✓	✓



NOVA's Key Features

- Features
  - High-performance
  - Strong Consistency
  - Snapshot support
  - Data protection

- Usage Models
  - open()/close(), read()/write()
  - DAX-mmap()

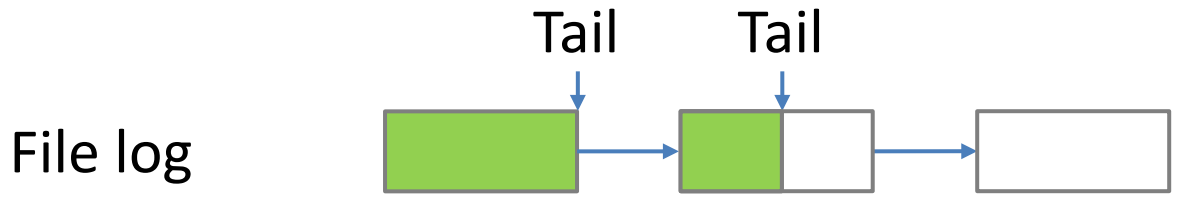


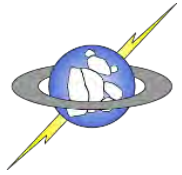
NOVA’s Architecture

Core NOVA Structures

Log Structure + copy-on-write + Journals

- One log per iNode
- Non-contiguous
- Fast, Simple atomic updates
- Meta-data only

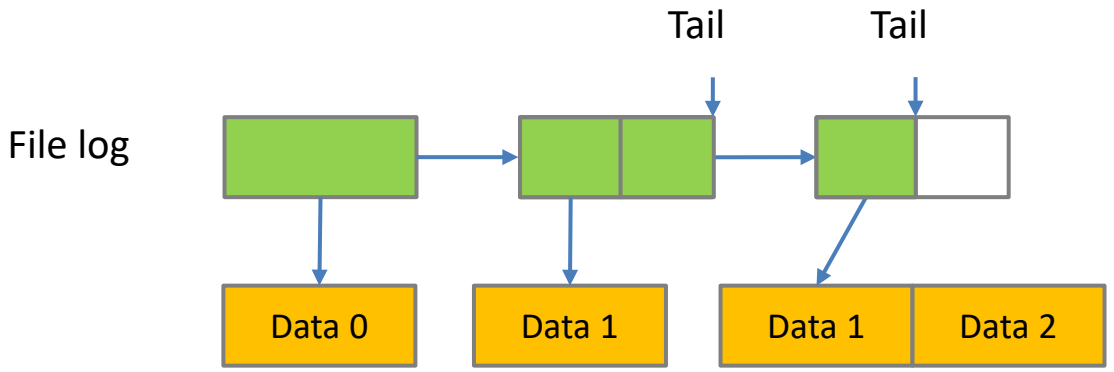


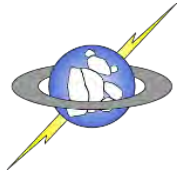


Core NOVA Structures

Log Structure + copy-on-write + Journals

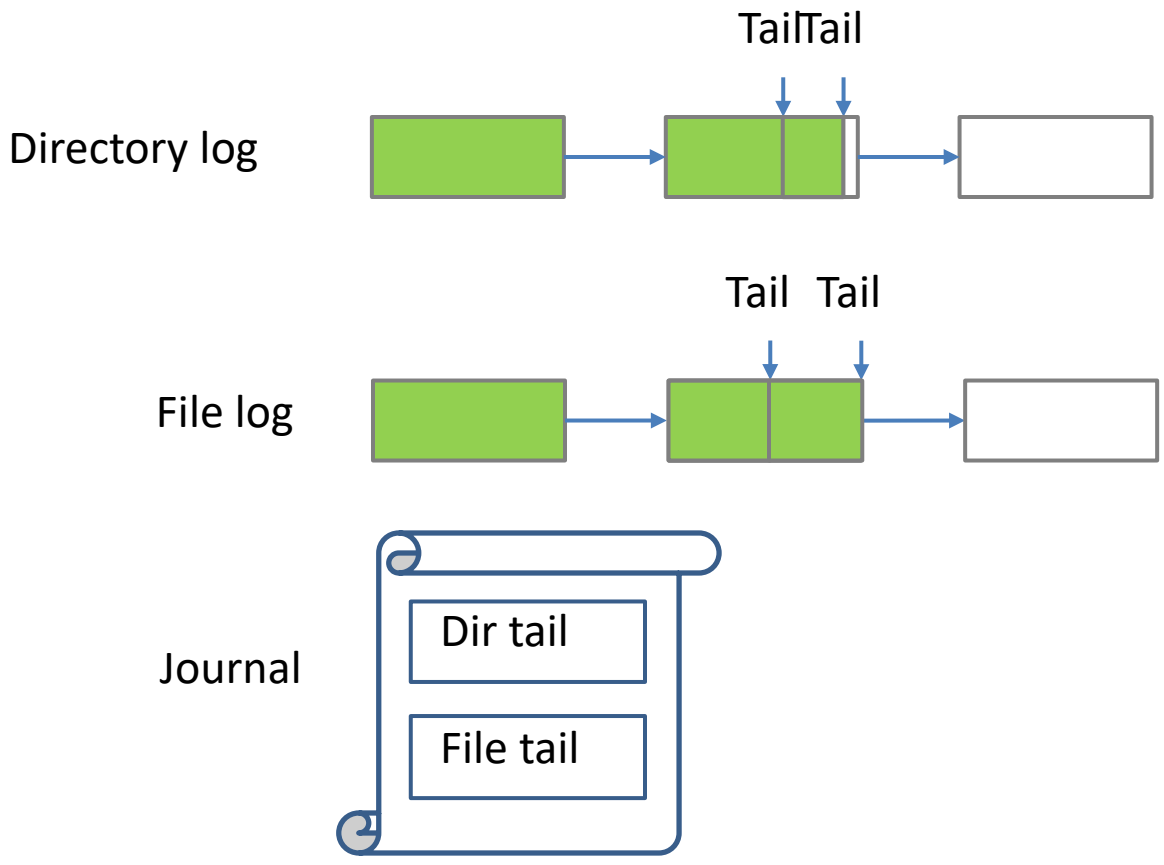
- Multi-page atomic update
- Fast allocation
- Instant data GC





Core NOVA Structures

Log Structure + copy-on-write + Journals

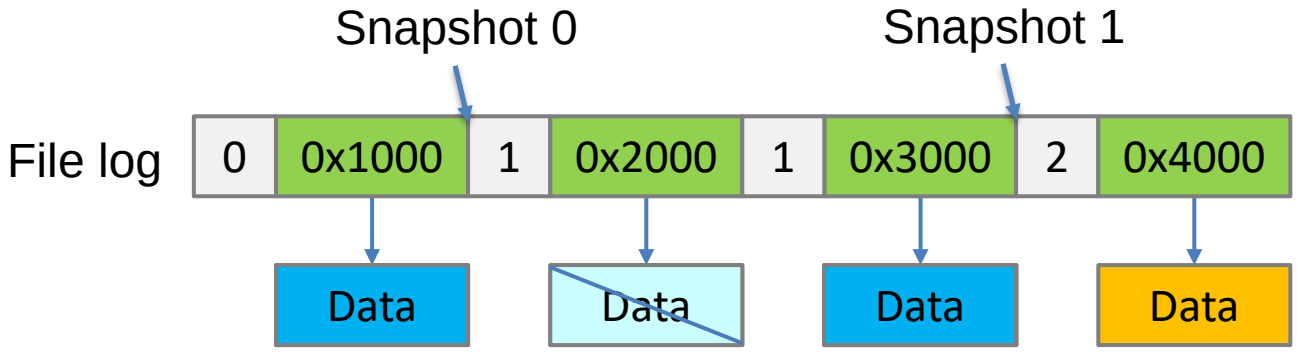


- Small, fixed sized journals
- For complex ops.

Supporting Backups with Snapshots

Snapshots for Normal File Access

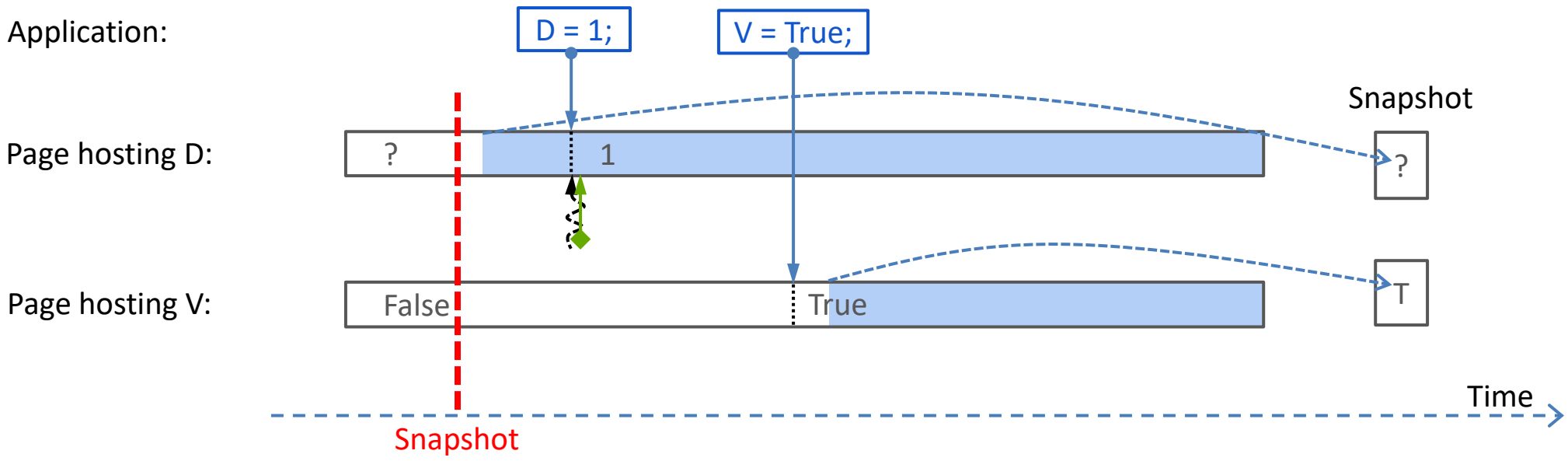
Current epoch 2



Snapshot entry
File write entry
Epoch ID
Data in snapshot
Reclaimed data
Current data

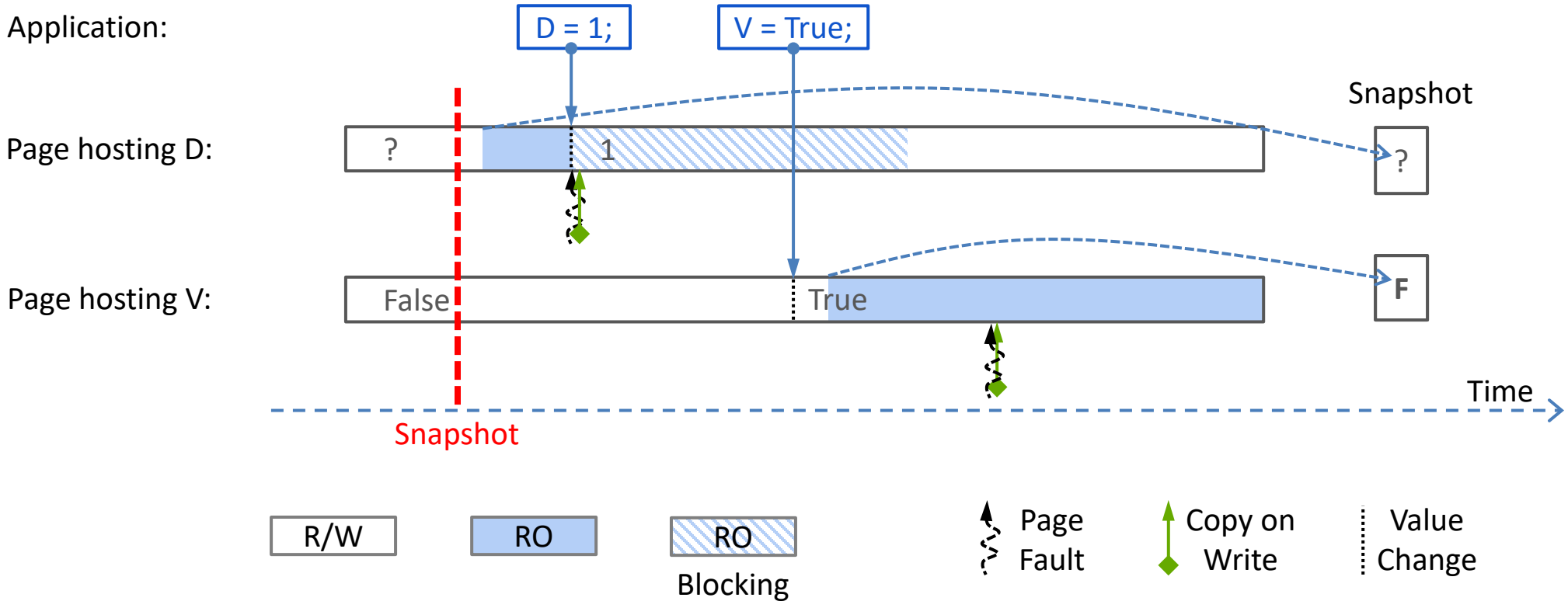
Corrupt Snapshots with DAX-mmap()

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



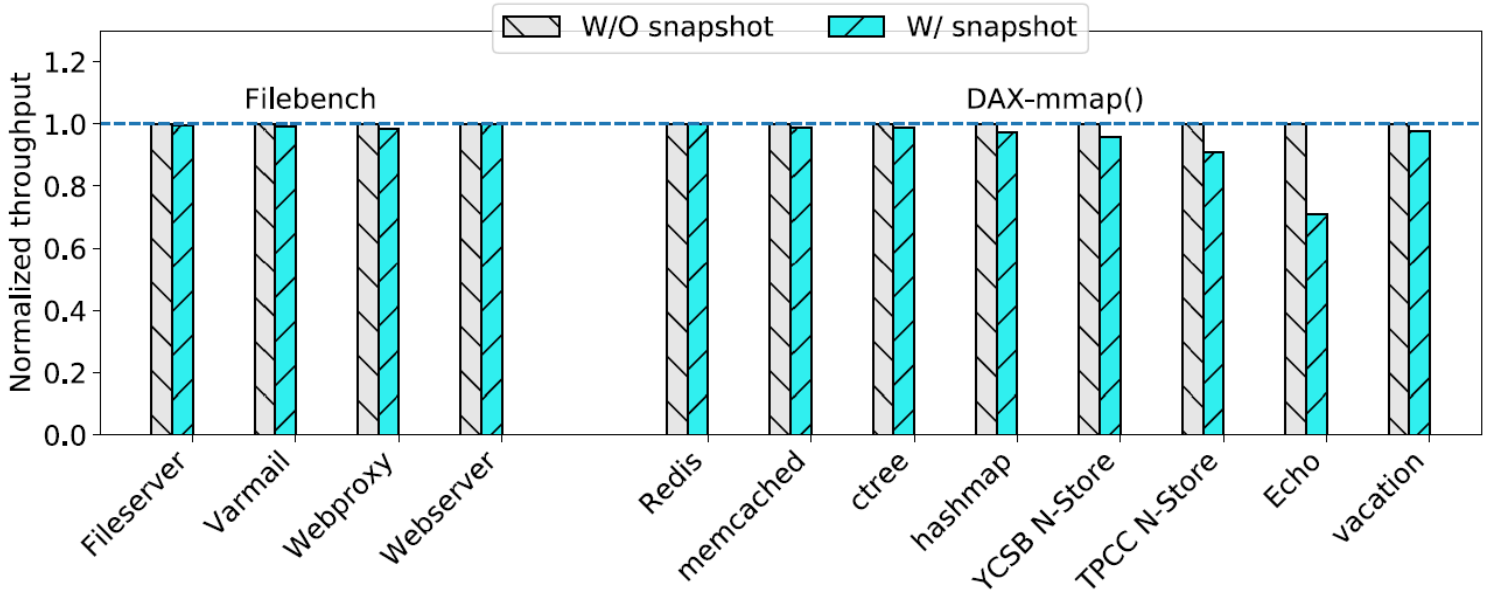
Consistent Snapshots with DAX-mmap()

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



Performance impact of snapshots

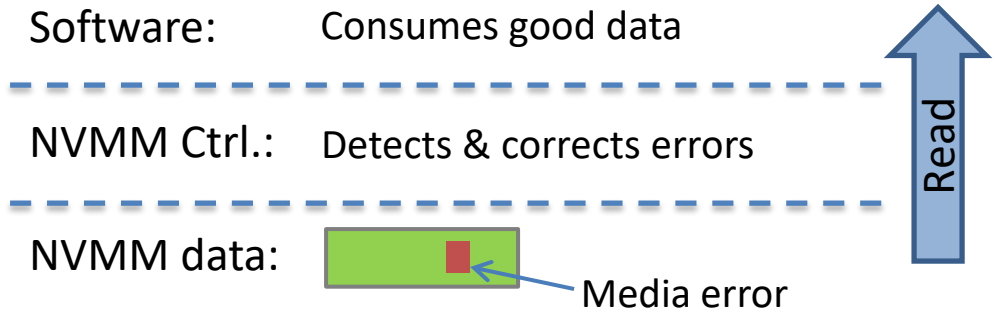
- Normal execution vs. taking snapshots every 10s
  - Negligible performance loss through read()/write()
  - Average performance loss 6.2% through mmap()

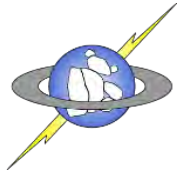


Data Protection: Metadata

NVMM Failure Modes: Media Failures

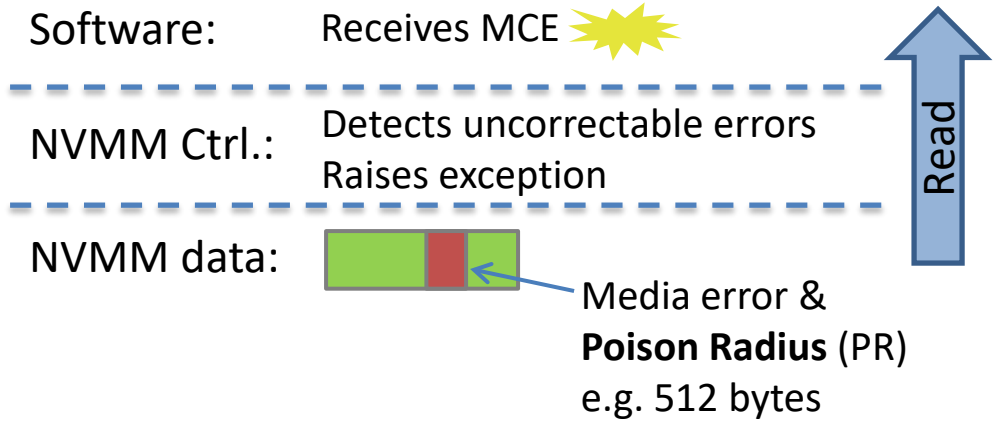
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - May consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

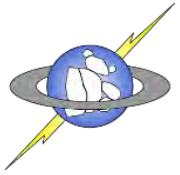




NVMM Failure Modes : Media Failures

- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - May consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

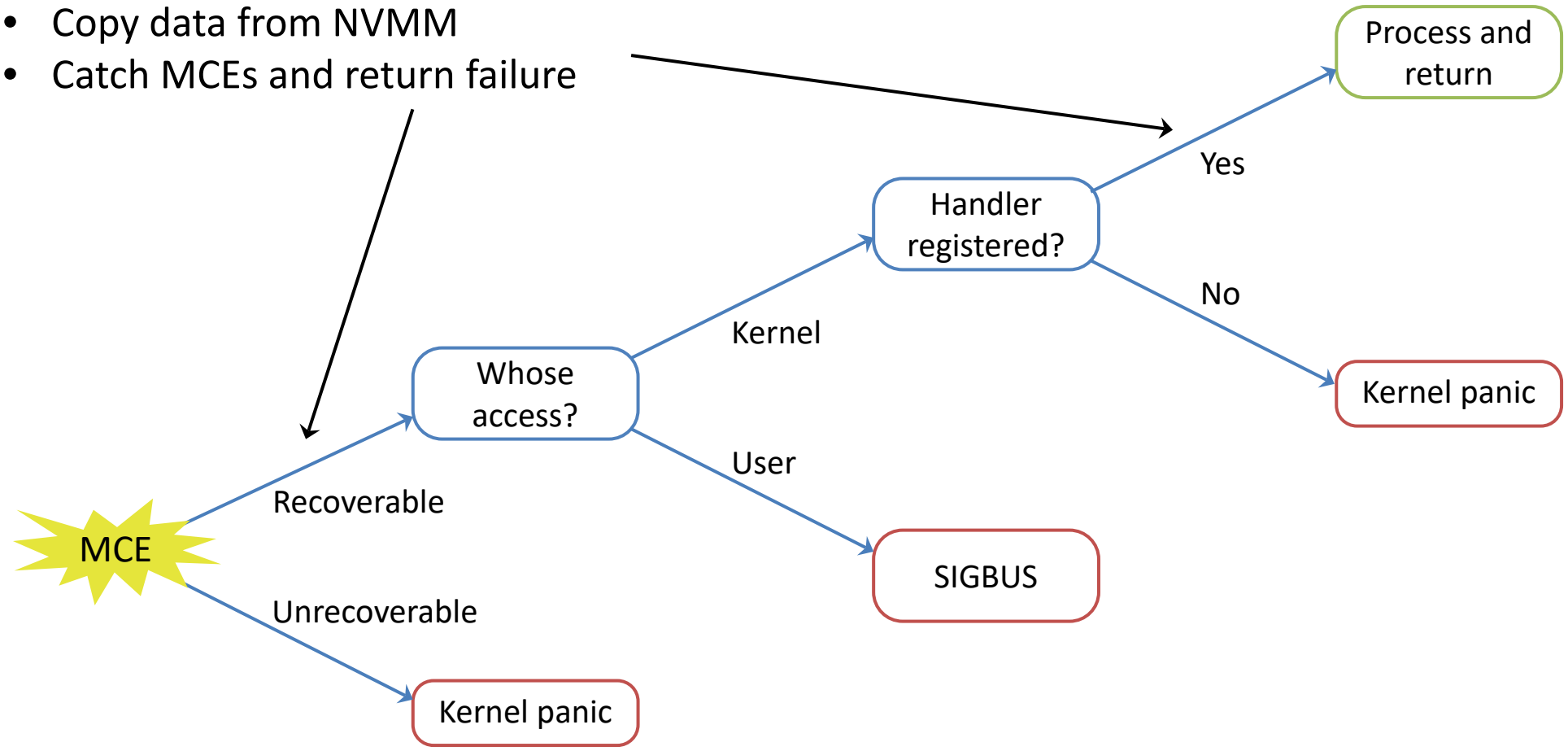


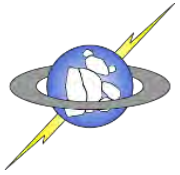


Detecting NVMM Media Errors

memcpy_mcsafe()

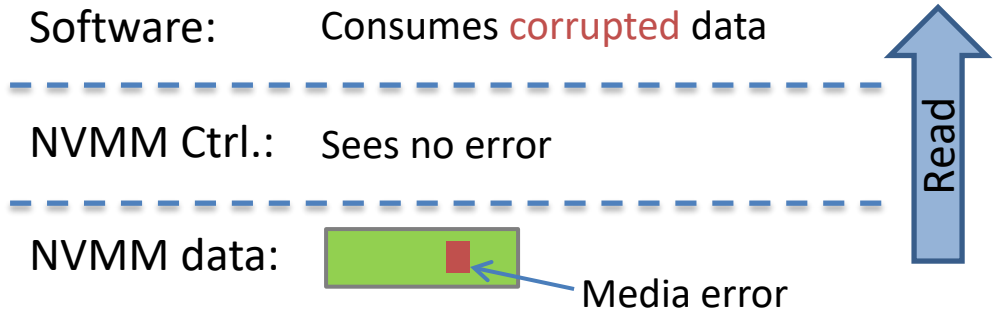
- Copy data from NVMM
- Catch MCEs and return failure

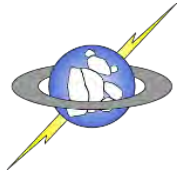




NVMM Failure Modes : Media Failures

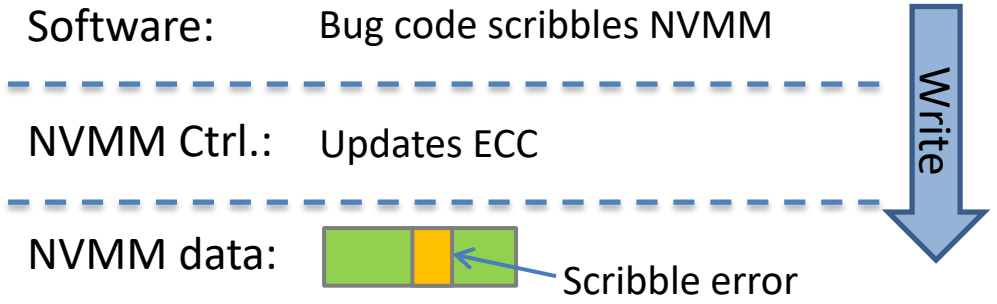
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

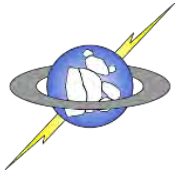




NVMM Failure Modes: Scribbles

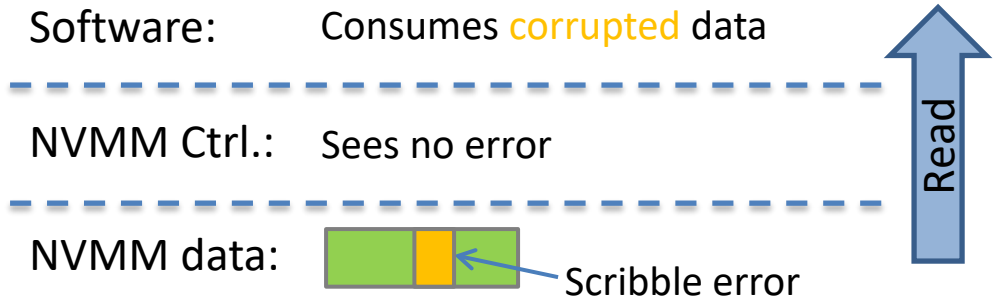
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software “scribbles”
  - Kernel bugs or NOVA bugs
  - NVMM file systems are highly vulnerable

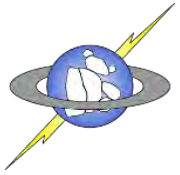




NVMM Failure Modes: Scribbles

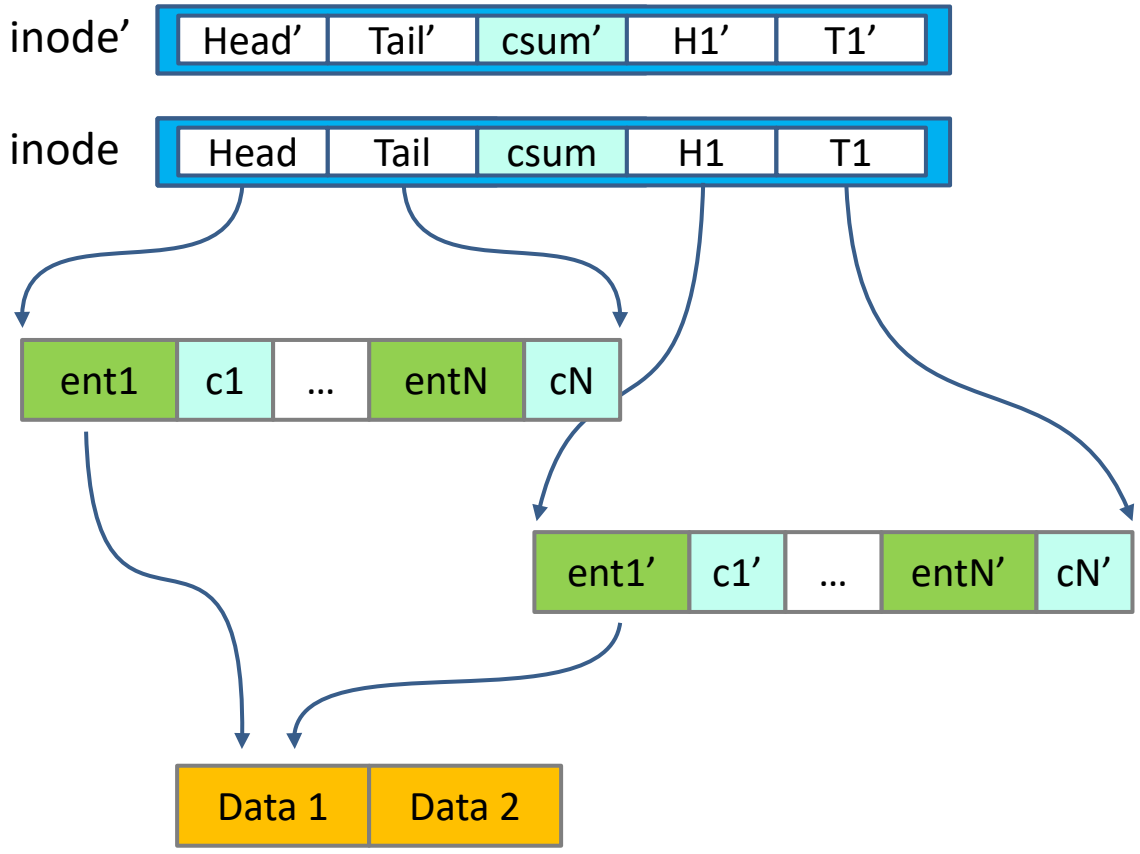
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software "scribbles"
  - Kernel bugs or NOVA bugs
  - NVMM file systems are highly vulnerable

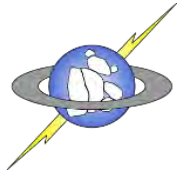




NOVA Metadata Protection

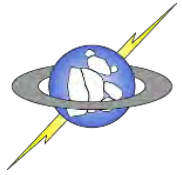
- Replicate everything
  - Inodes
  - Logs
  - Superblock
  - …
- CRC32 Checksums everywhere





Defense Against Scribbles

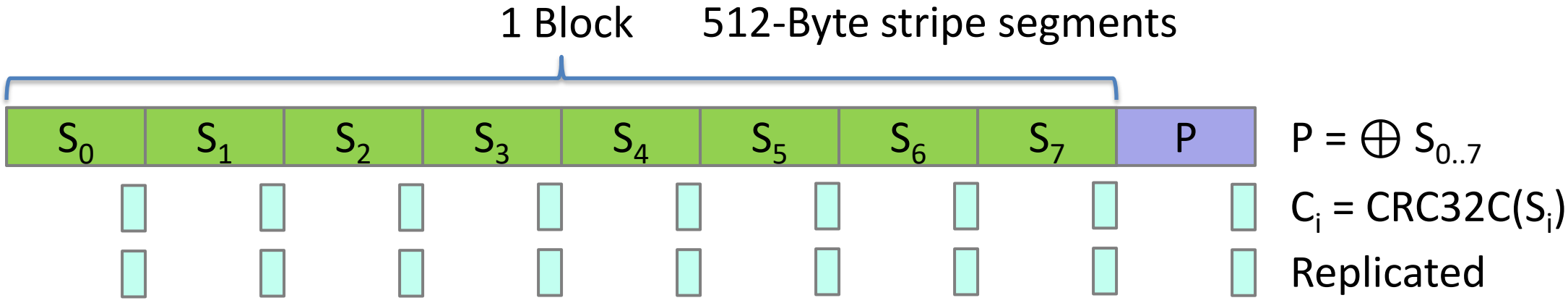
- Tolerating Larger Scribbles
  - Allocate replicas far from one another
  - Can tolerate arbitrarily large scribbles to metadata.
- Preventing scribbles
  - Mark all NVMM as read-only
  - Disable CPU write protection while accessing NVMM

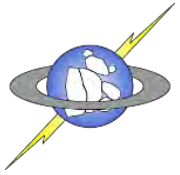


Data Protection: Data

NOVA Data Protection

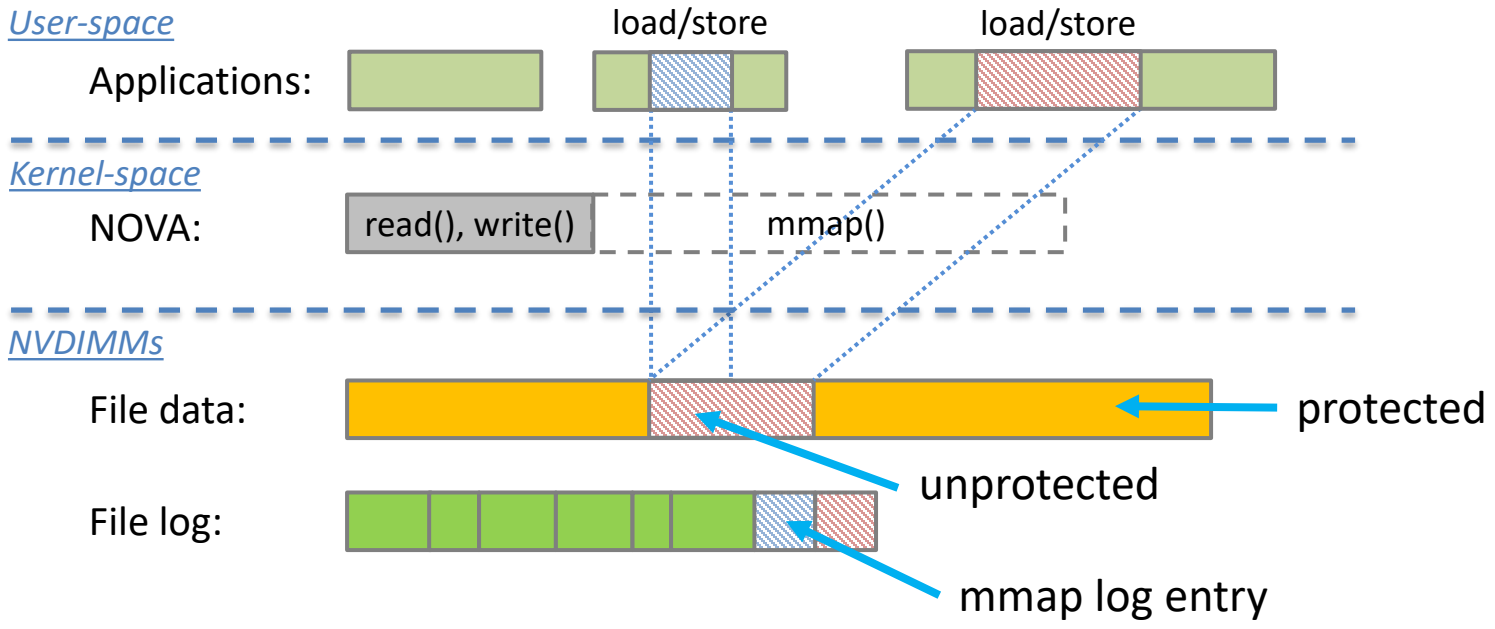
- Divide 4KB blocks into 512-byte stripes
- Compute a RAID 5-style parity stripe
- Compute and replicate checksums for each stripe

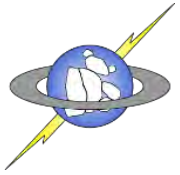




File data protection with DAX-mmap

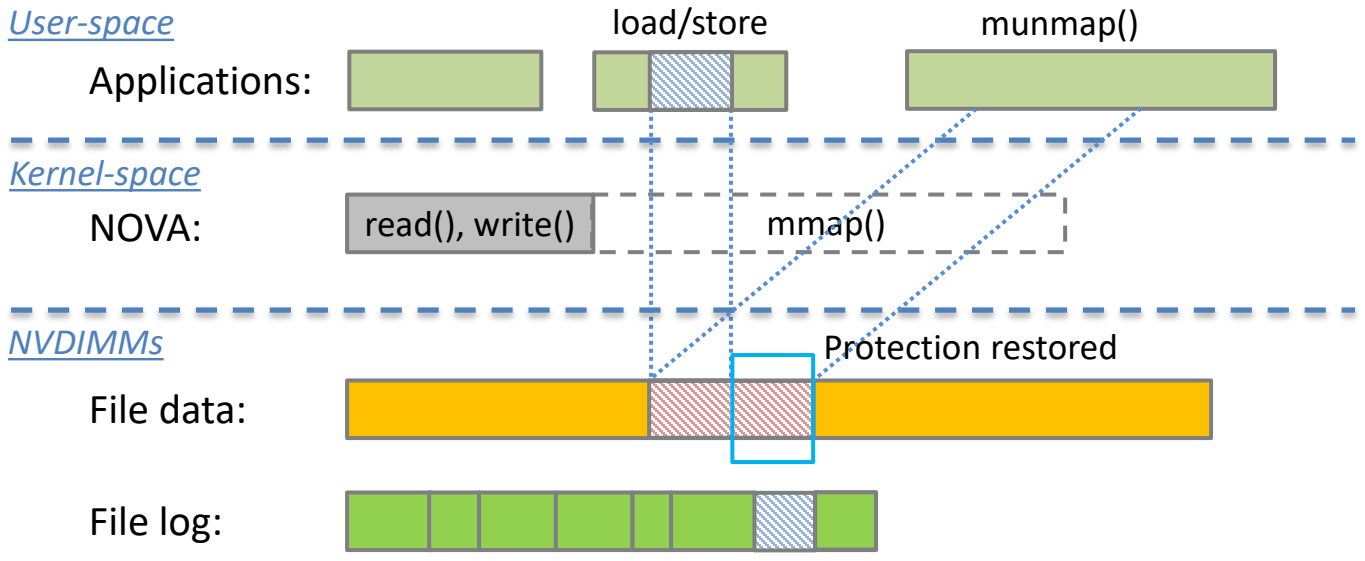
- With DAX-Mmap(), file data changes are invisible to NOVA
- NOVA cannot protect mmap'ed file data
- NOVA logs mmap() and restores protection on munmap() or recovery





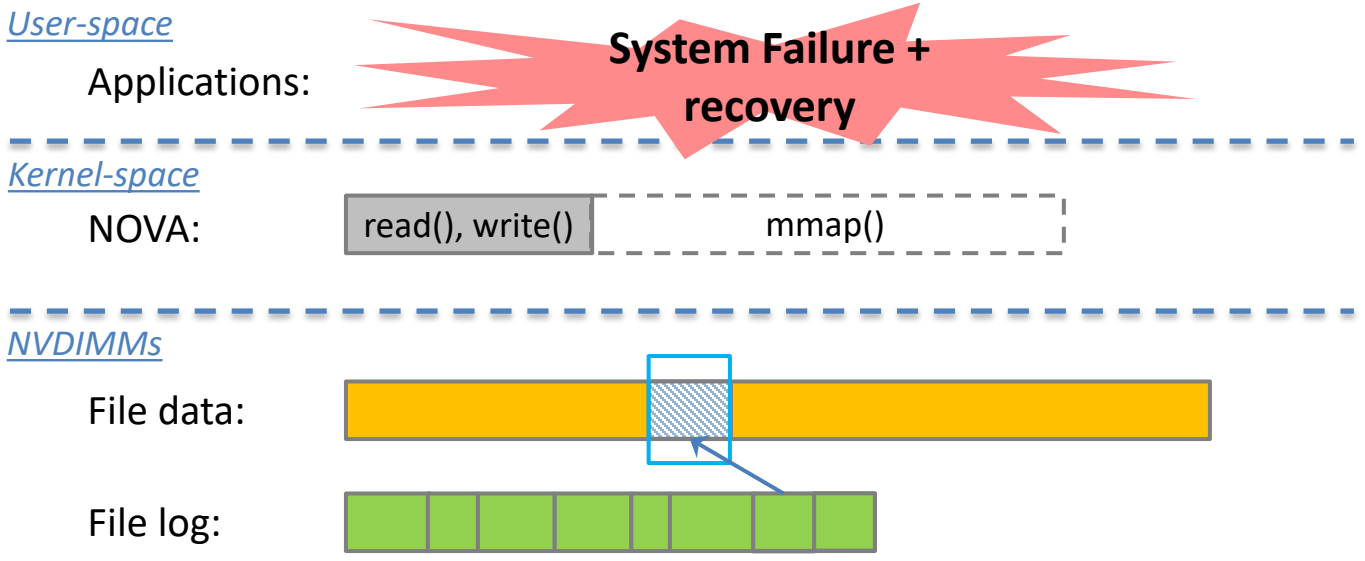
File data protection with DAX-mmap

- NOVA cannot protect mmap'ed file data
  - User applications directly load/store the mmap'ed region
  - NOVA has to know what file pages are mmap'ed



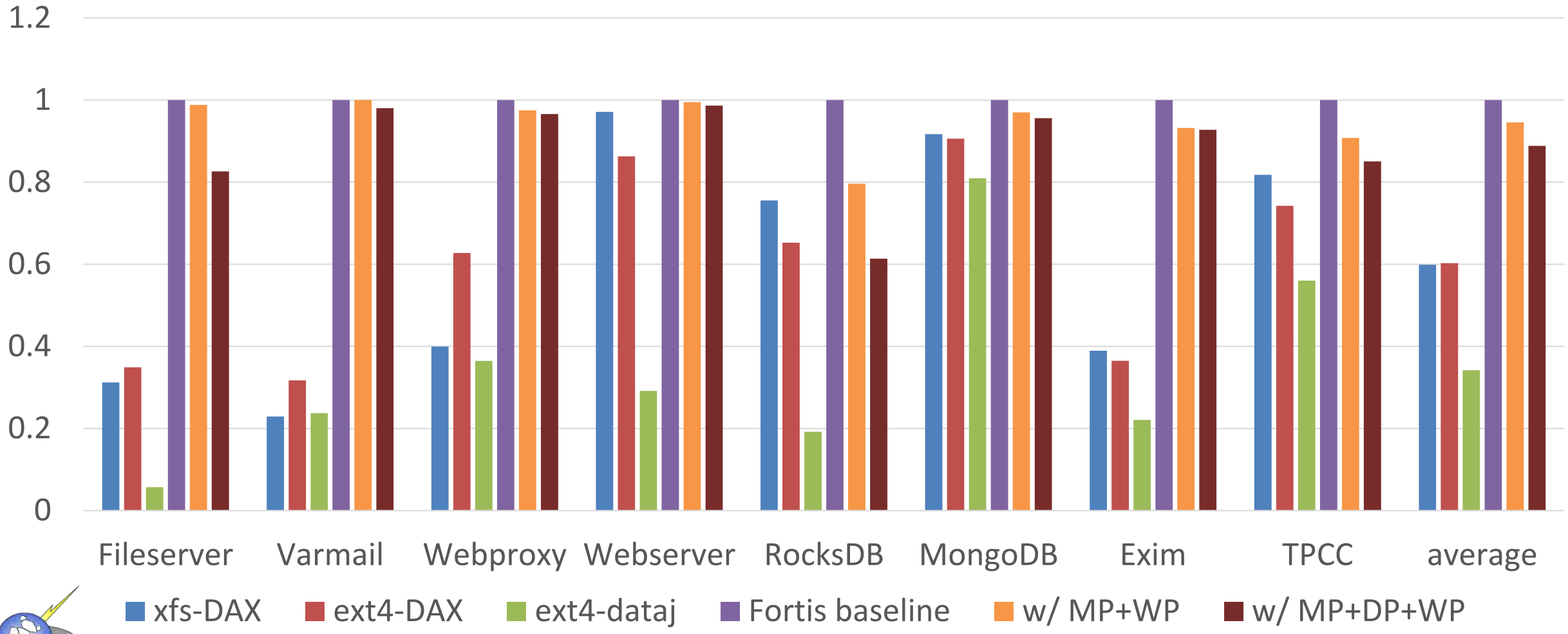
File data protection with DAX-mmap

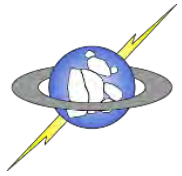
- NOVA cannot protect mmap'ed file data
  - User applications directly load/store the mmap'ed region
  - NOVA has to know what file pages are mmap'ed



Performance

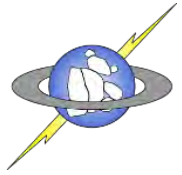
Performance Cost of Data Integrity





Conclusion

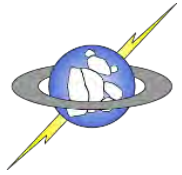
- Existing file systems do not meet the requirements of applications on NVMM file systems
- NOVA's multi-log design achieves high performance and strong consistency
- NOVA's data protection features ensure data integrity
- NOVA outperforms existing file systems while providing stronger consistency and data protection guarantees



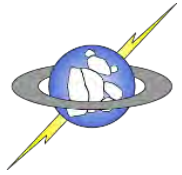
Thank you!

Try NOVA!

https://github.com/NVSL/NOVA

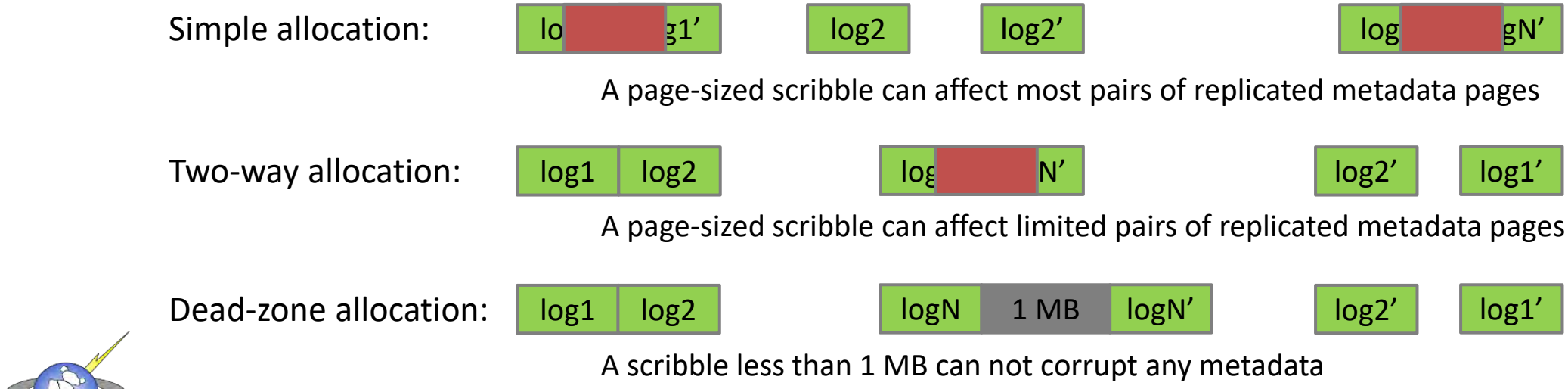


Backup Slides



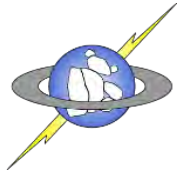
Protecting Against Scribbles

- Metadata allocator separates metadata replicas
  - Allocate primary and replica pages in opposite directions
  - Use allocator ‘dead-zone’ to guarantee minimal distance
  - Protect against scribbles from other kernel bugs and own bugs

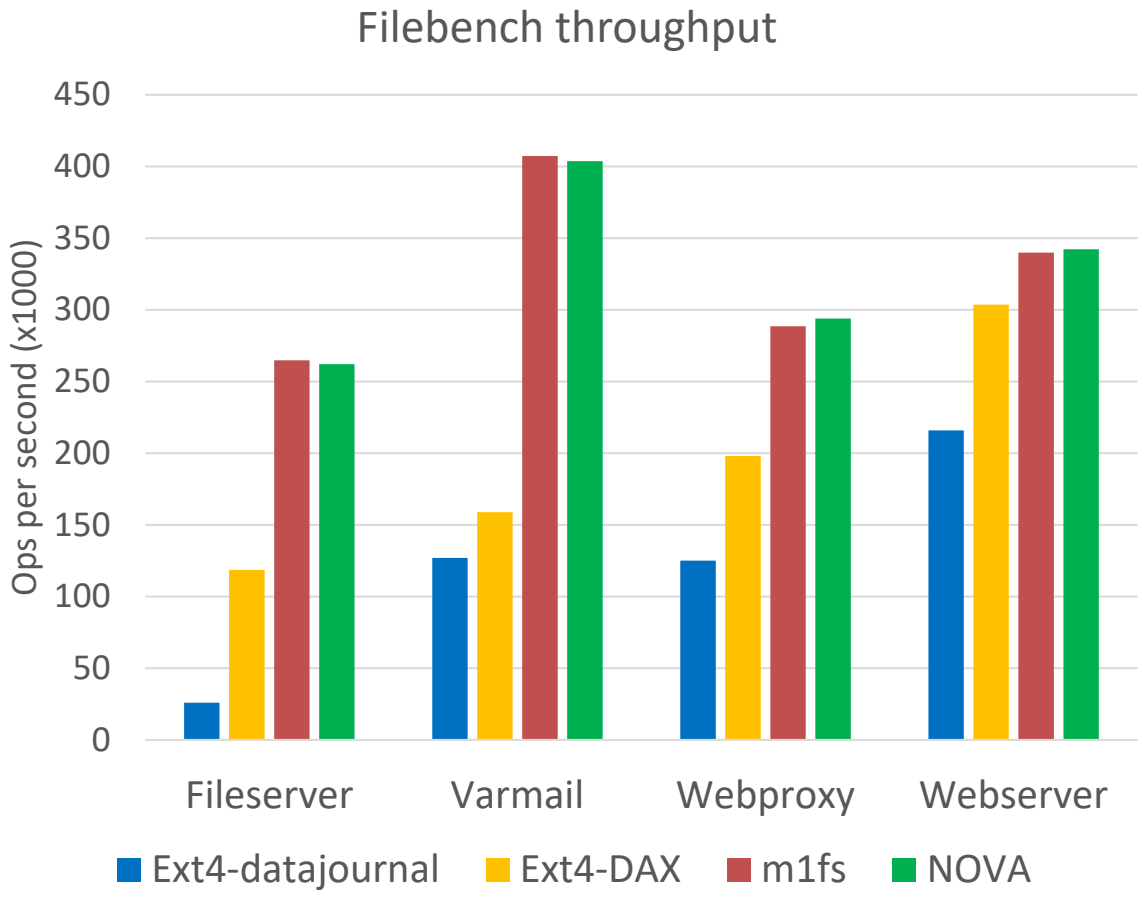


Minimize the chance of corruptions – x86 write protection

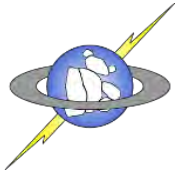
- Leverage x86 CPU's write protection
  - CR0.WP disables/enables writing to RO memories of each x86 core
  - Only enable writing when NOVA writes to NVMM
  - Protect against scribbles from other kernel bugs, not own bugs



Filebench throughput

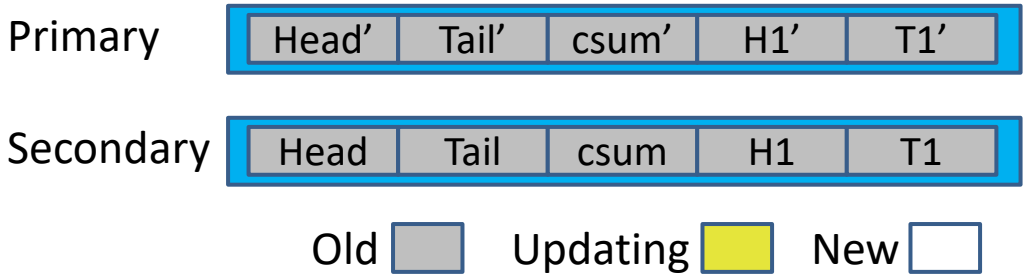


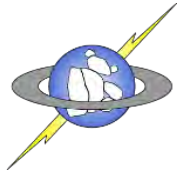
- NOVA achieves high performance with strong data consistency



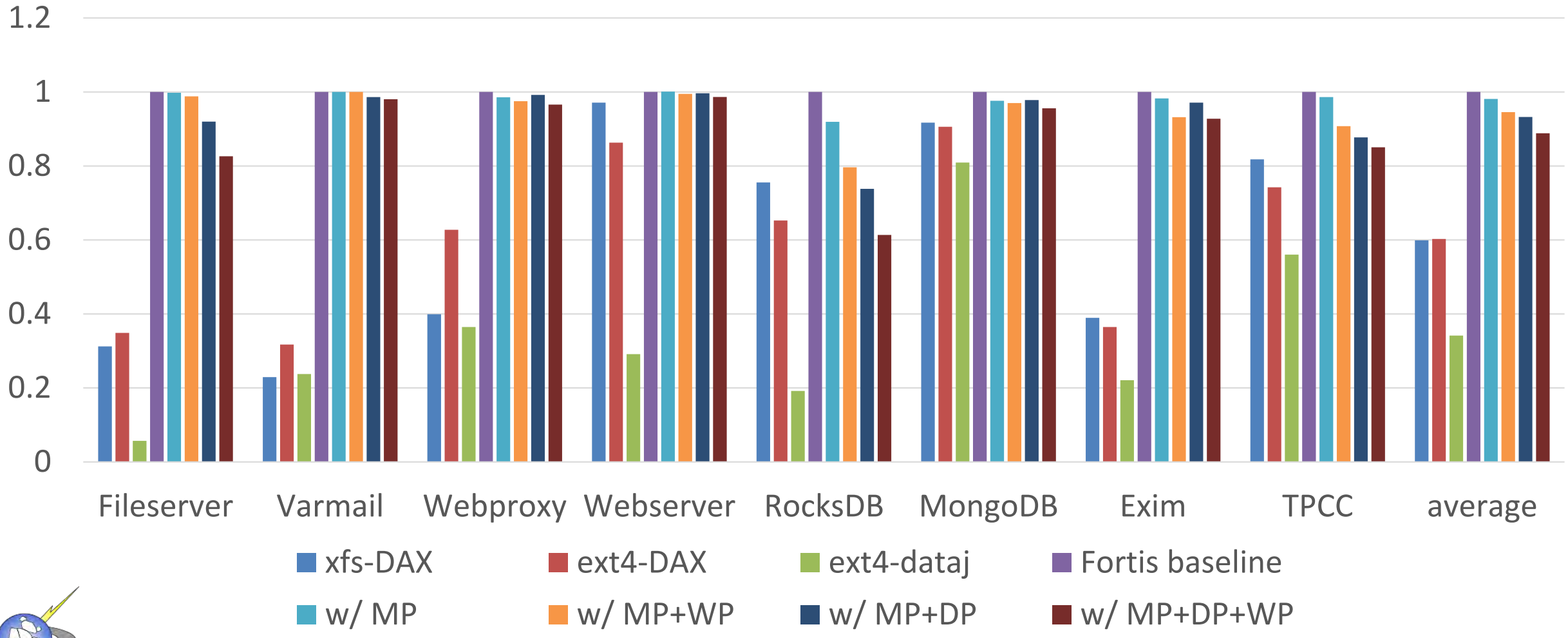
Tick-tock inode update

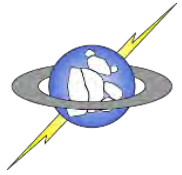
- Update tails of primary inode
- Update csum of primary inode
- Same procedure for inode’





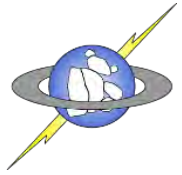
Performance Cost of Data Integrity





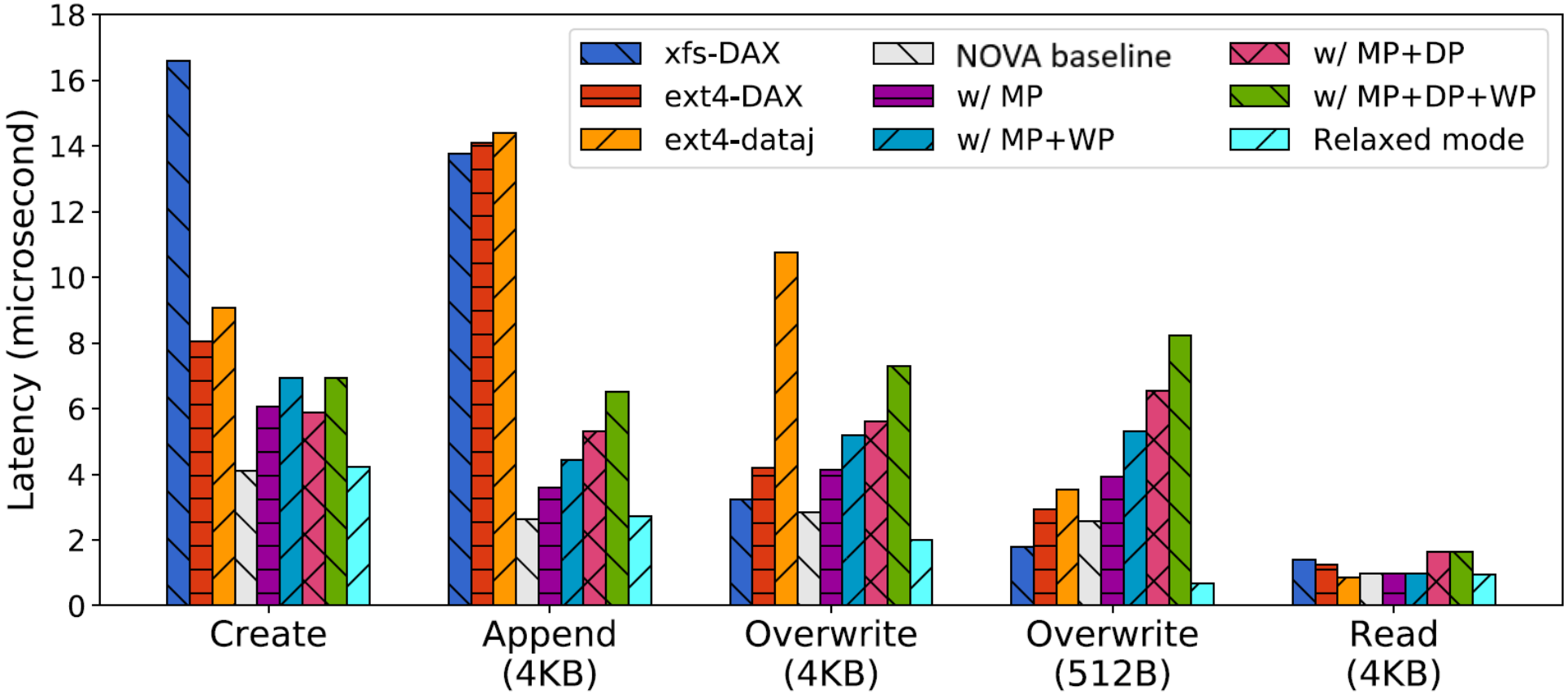
Conclusions

- NVMM file systems need unique solutions for reliability
  - Error reporting mechanisms different than disks
  - DAX-mmap complicates designs
- Performance and storage penalties vary
  - Storage cost is modest for the presented hardening techniques
  - Performance impact is significant for some applications
- More knowledge is necessary to determine the trade-offs
  - Uncorrectable media errors in emerging NVMM technologies
  - The frequency and size of scribbles in kernel space
- NOVA provides all hardening techniques as mount options



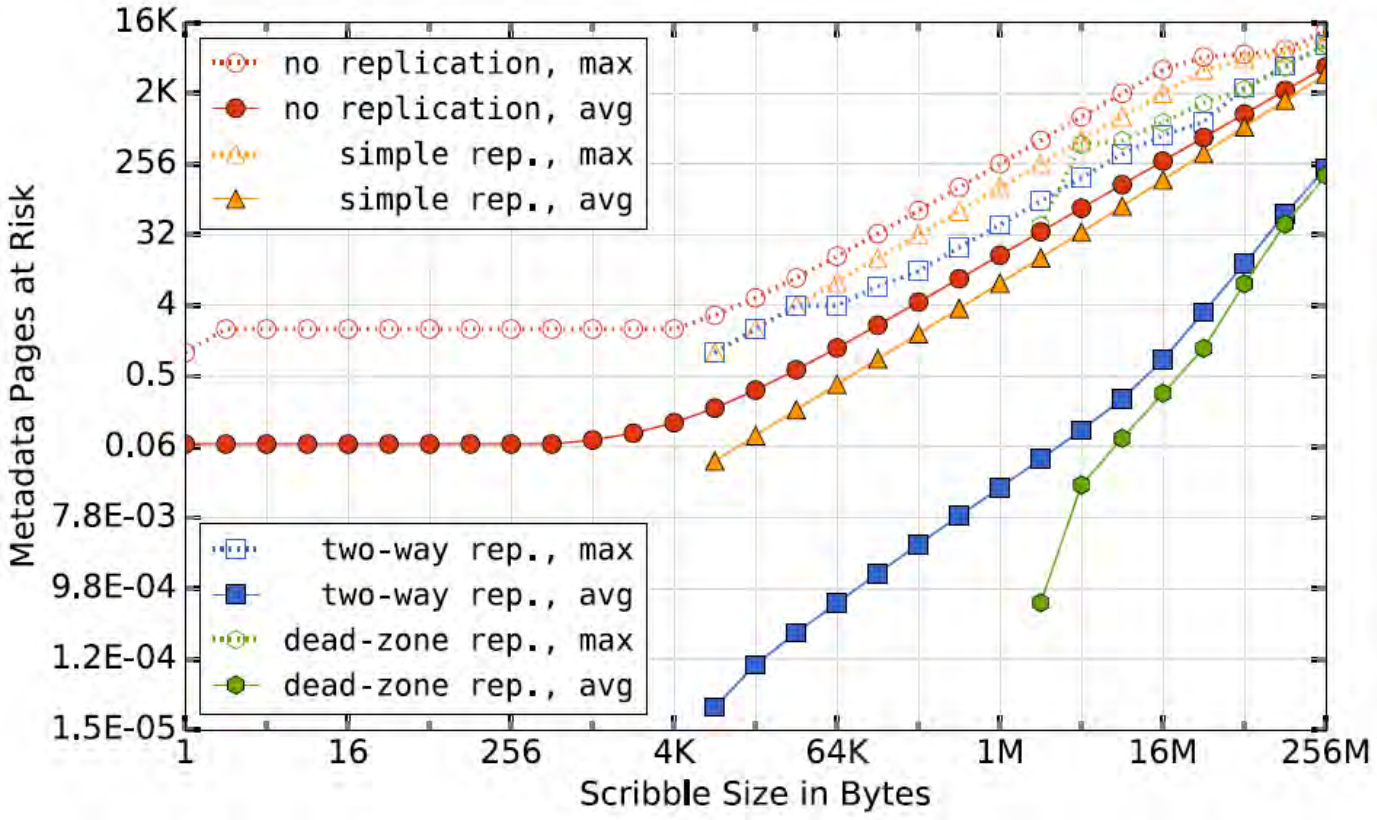
Performance impact of data integrity

- File operation latency



Reliability evaluation – metadata pages at risk

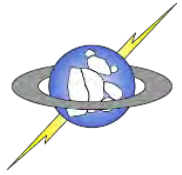
- Scan an aged NOVA file system image
  - Examine distances between the primary and replica pages
  - Count vulnerable page pairs for a given scribble size



Y == 0 points do not show in log-log plot

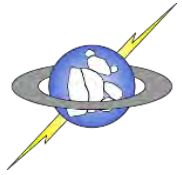
PMFS shortcomings

- No data atomicity support
- High consistency overhead with persistent B-tree
- Not scalable
  - Directory operations (linear search)
  - NVMM allocation (Single allocator)
  - Single journal shared by all transactions
- Poor performance on large directories
- Intel has deprecated PMFS

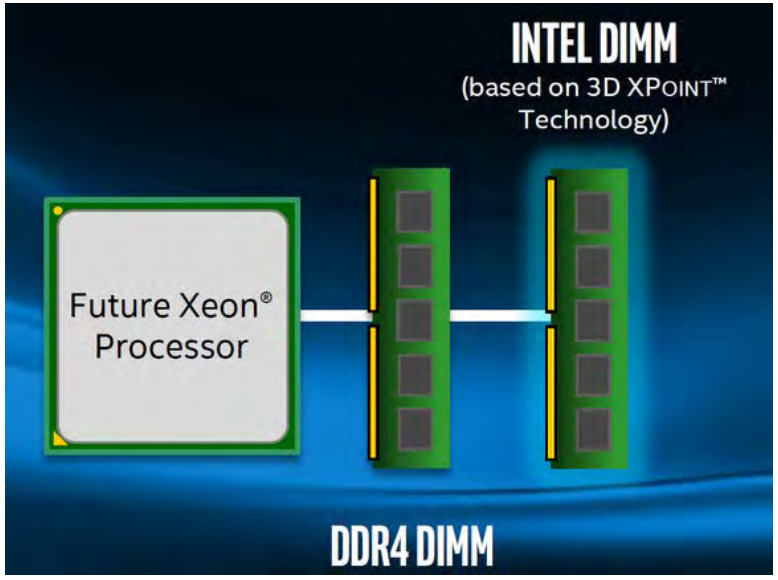


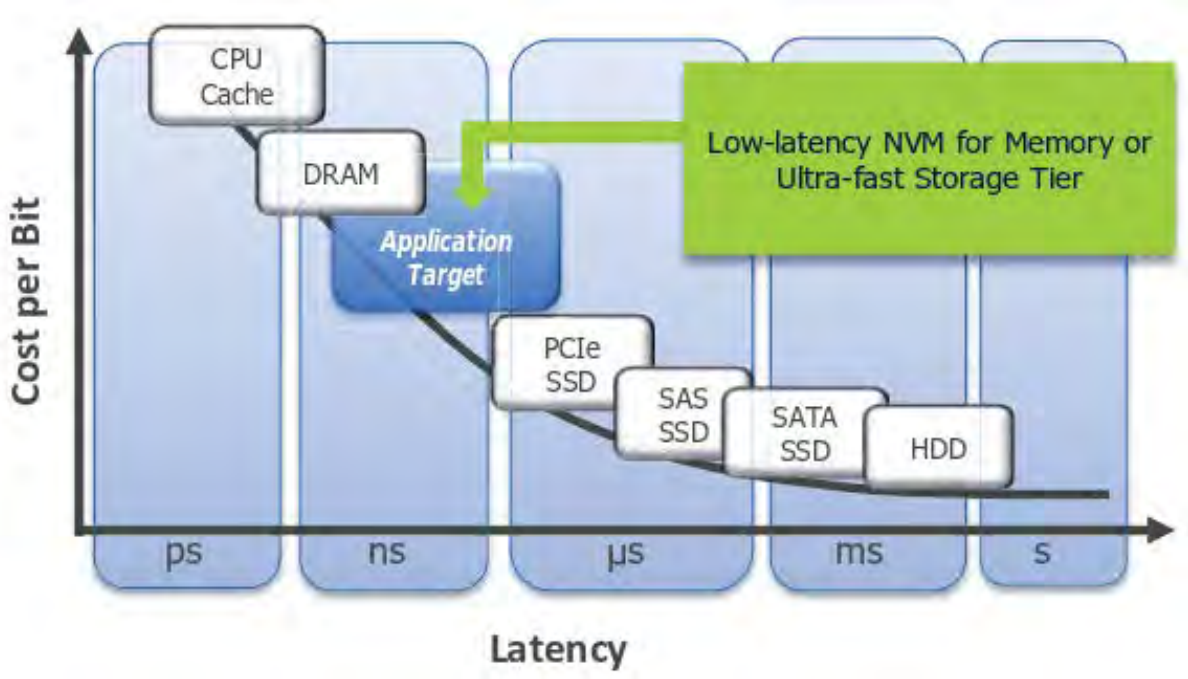
Ext4-DAX and xfs-DAX shortcomings

- No data atomicity support
- Single journal shared by all the transactions (JBD2-based)
- Poor performance

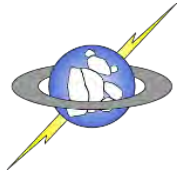


Non-volatile main memory is about to happen

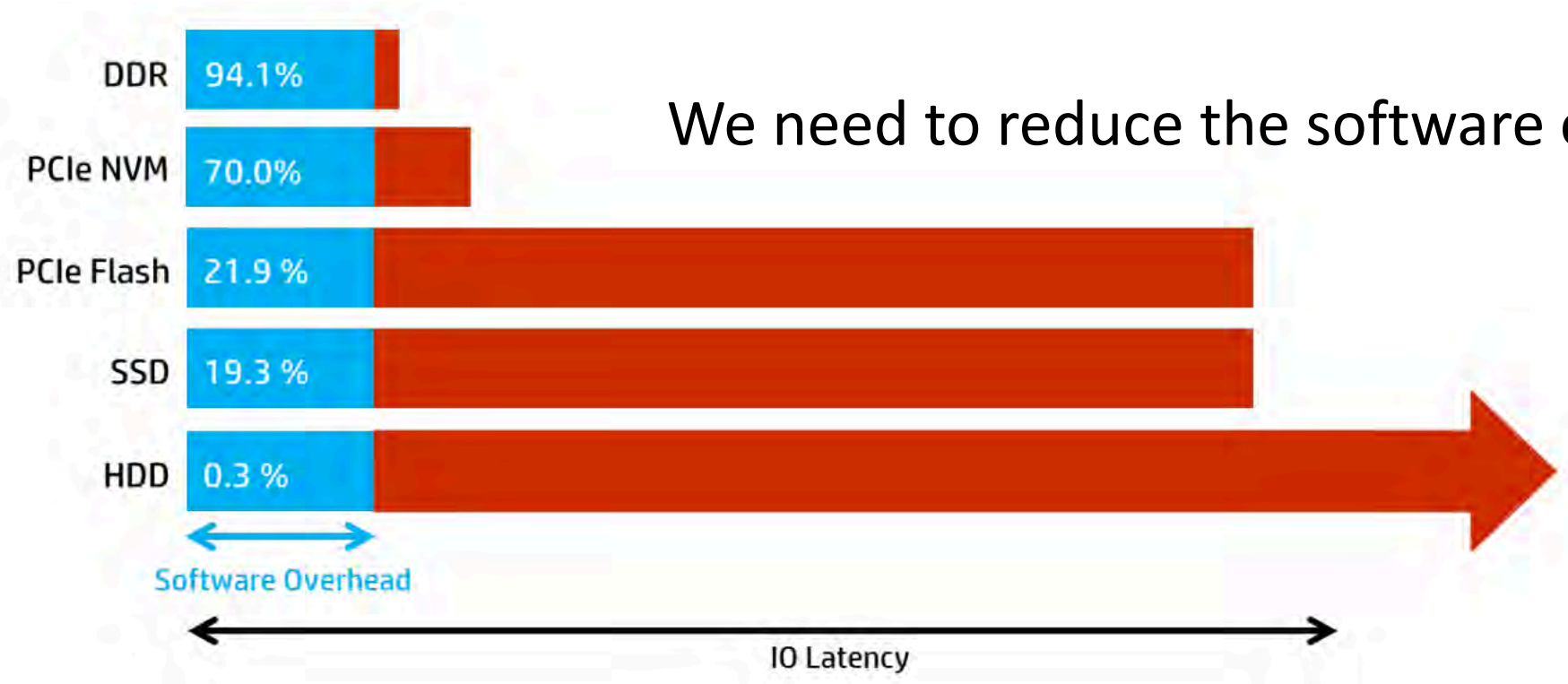




NVMM needs a new file system: PMFS, Ext4-DAX, SCMFS, Aerie, NOVA, …



Why a new file system?



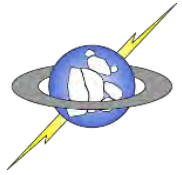
We need to reduce the software overhead.

Source: Memory-Driven Computing, Kimberly Keeton, HP Labs

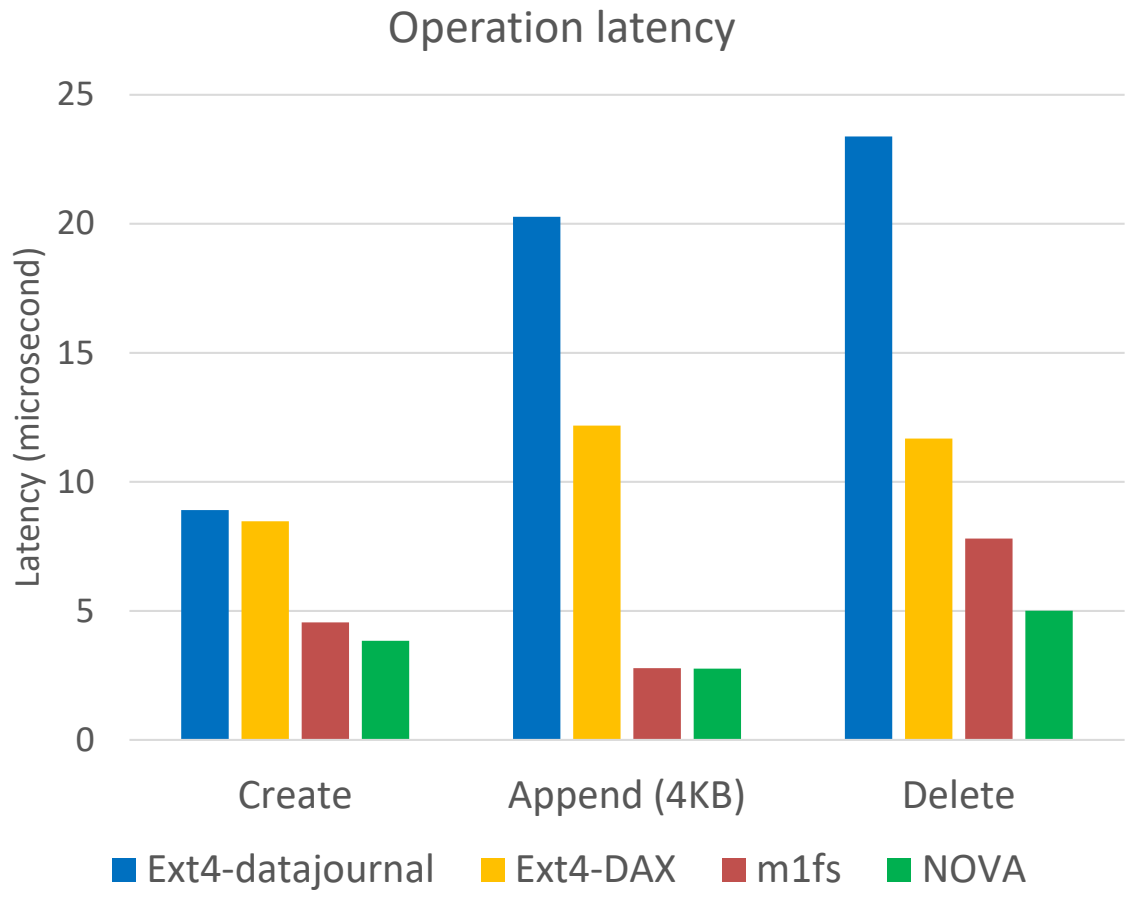
What Should a File System Provide?

- Performance ⟵ current focus (of all known efforts)
- Consistency
  - Atomic metadata operations
  - Atomic file updates
- Data Protection
  - Snapshots
  - Media error protection
  - Software error protection
- Cost optimizations
  - Compression
  - Deduplication

We need to study the impact of adding more file system services in the context of NVMM.



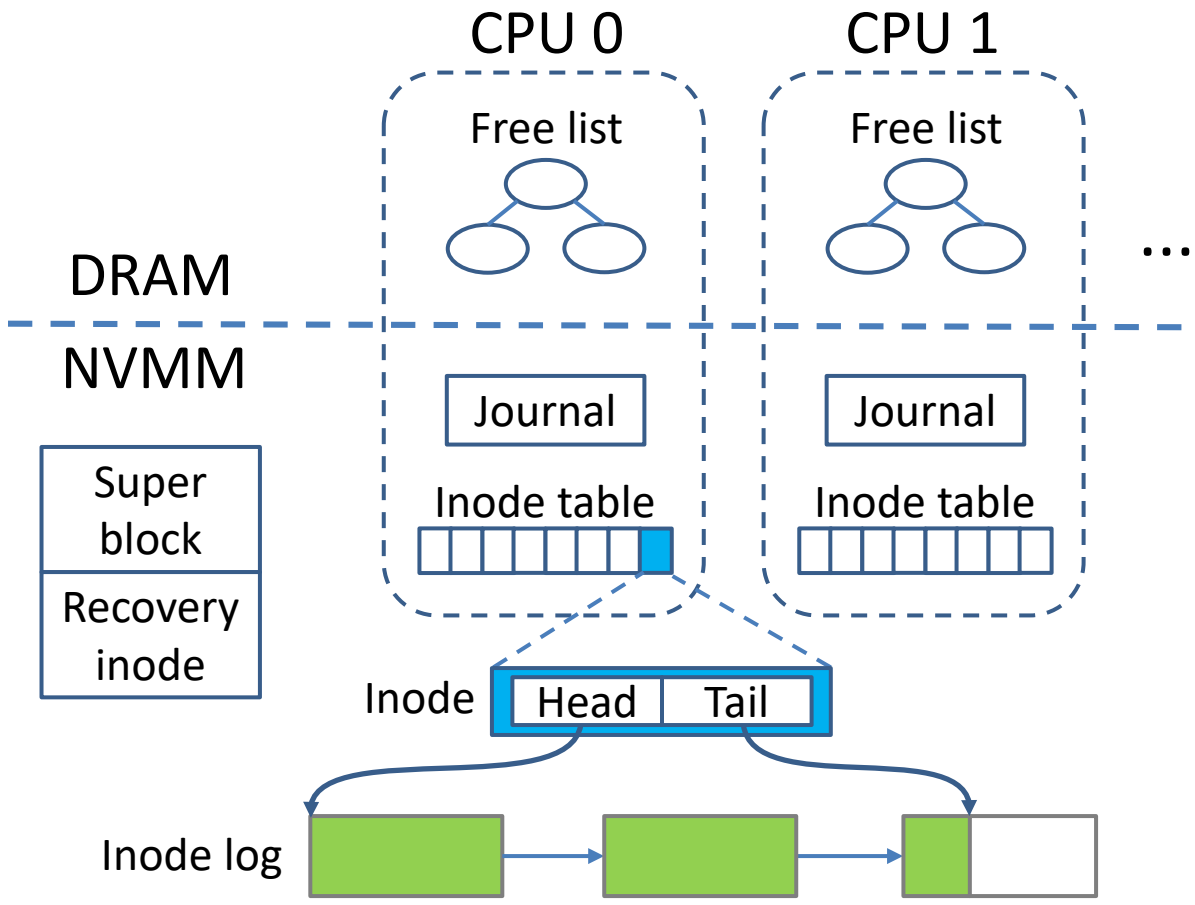
Evaluation: Latency

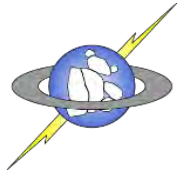


- Intel PM Emulation Platform
  - Emulates different NVM characteristics
  - Emulates clwb/PCOMMIT latency
- NOVA provides low latency atomicity

NOVA design and in-NVMM data layout

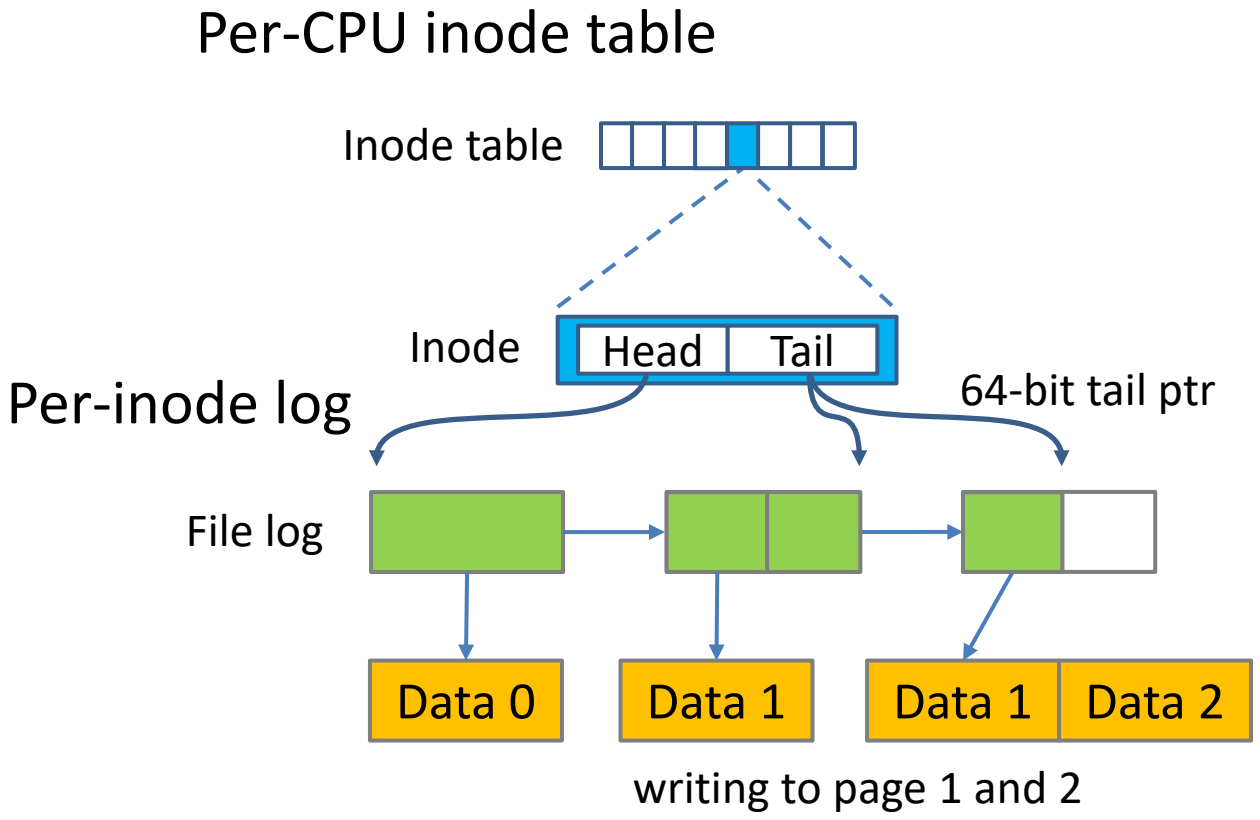
- High performance
  - No page cache
  - Memory semantics
  - Segregated data structures
  - Per-CPU freelist
  - Per-inode logging
- Strong consistency
  - Copy-on-write file data
  - Using 8-byte atomic stores

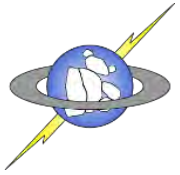




NOVA design and in-NVMM data layout

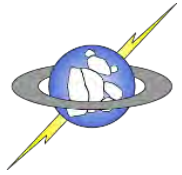
- High performance
  - No page cache
  - Memory semantics
  - Segregated data structures
  - Per-CPU freelist
  - Per-inode logging
- Strong consistency
  - Copy-on-write file data
  - Using 8-byte atomic stores





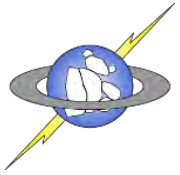
NVMM file systems should support snapshot

- Snapshot is essential for file system backup
- Available in file systems for block devices
  - ZFS, Btrfs, WAFL
- NOVA is the first NVMM file system providing snapshot
  - Efficient full-filesystem snapshot at minimal performance cost
  - Creating/deleting snapshots does not halt file system
  - Creating consistent snapshots with DAX-mmap enabled

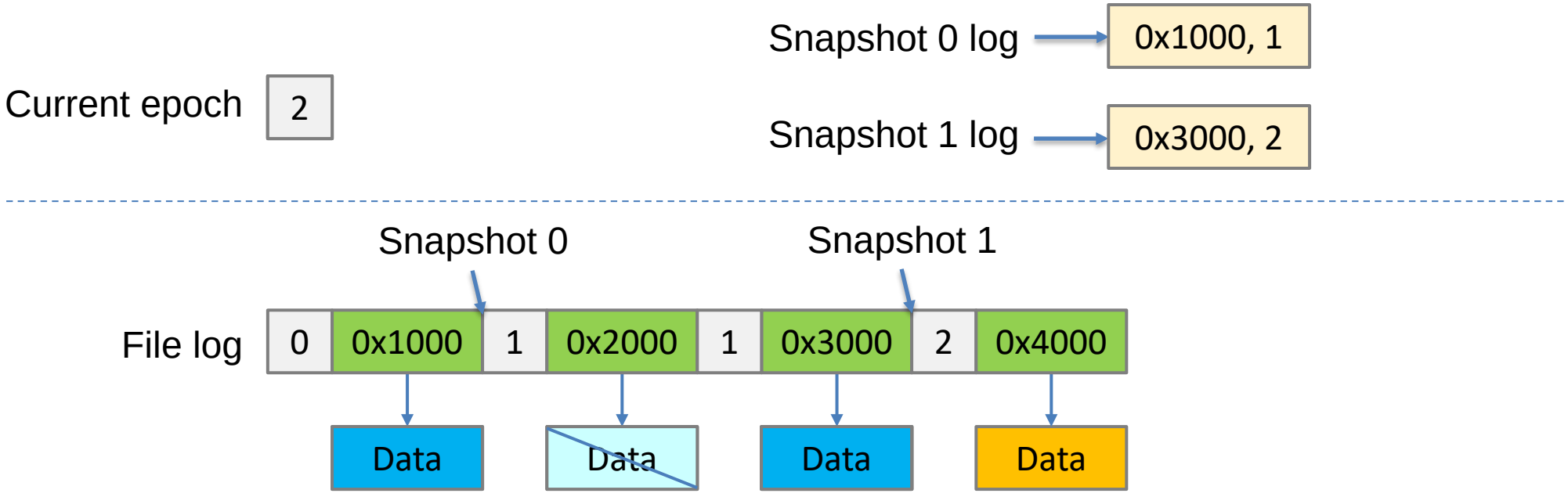


Enable snapshot in NOVA

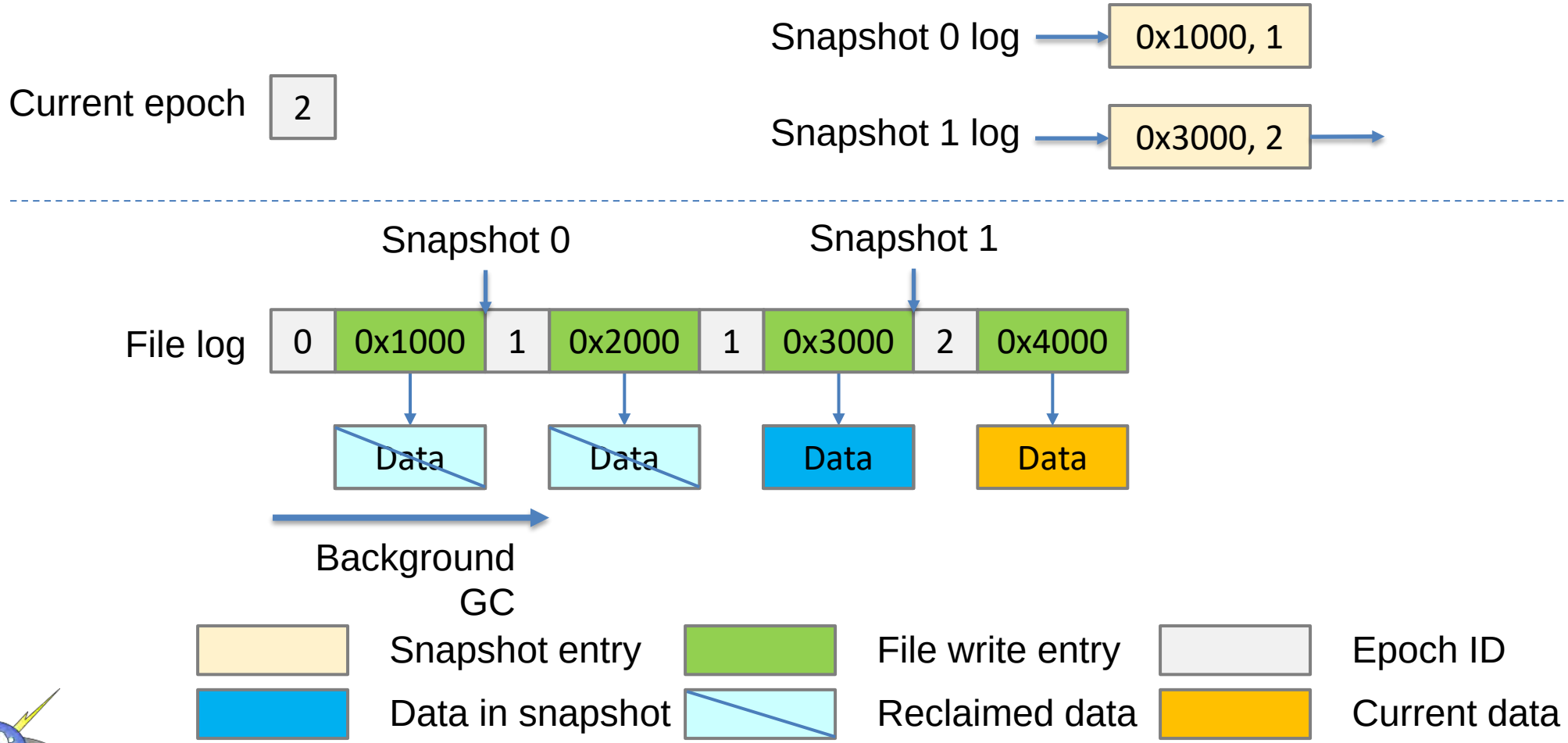
- Maintain a current ‘epoch_id’ for the file system
  - Stored in the superblock
  - Incremented after every snapshot taken
- Add the ‘epoch_id’ to each log entry



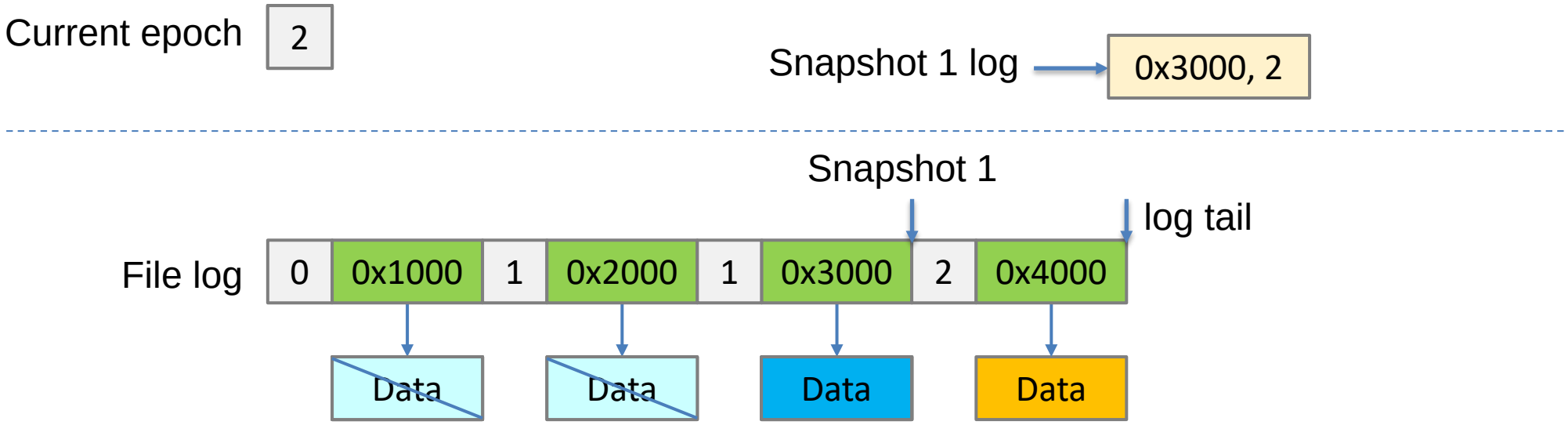
Taking snapshots

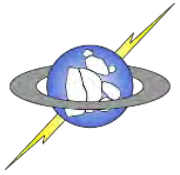


Deleting snapshots



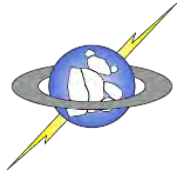
Mounting snapshots





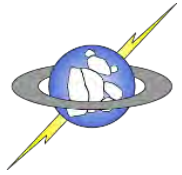
Snapshots with DAX-mmap

- Goal: Applications take snapshots and keep running
  - Virtual addresses do not change
  - Consistency must be guaranteed
- How: Set each mmap'ed page as read-only
  - Then do copy-on-write for new stores (detected by page fault)
- Caveat: Can only atomically set one page as read-only
  - What if the order of becoming read-only conflicts with consistency?



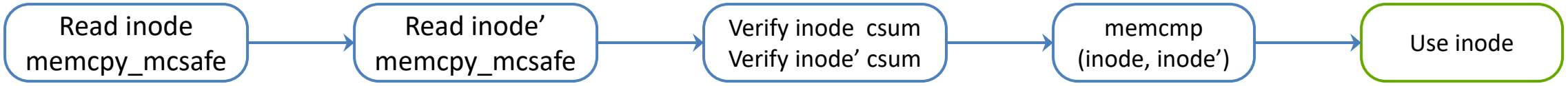
NOVA (meta)data integrity features

- Detect (meta)data corruptions
  - Media errors: error code from memcpy_mcsafe(), and checksums
  - Software scribbles: checksums
- Repair (meta)data corruptions
  - Metadata: Fully replicated
  - File data: Stripe and parity-code each block
- Minimize scribbles
  - Leverage x86 CPU's write protection (CR0.WP)
  - Metadata allocators separate replicas

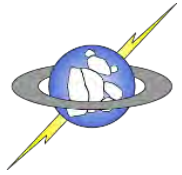


Metadata error detection and correction

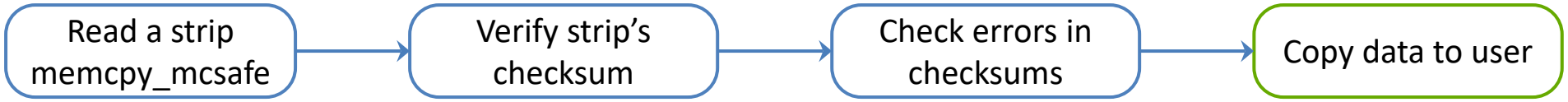
- Use inode access as an example:



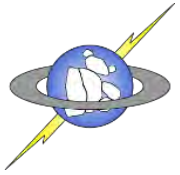
- If any step raises an error:
  - Attempt to repair and retry
  - If recovery fails: return `-EIO` to user



File data error detection and correction

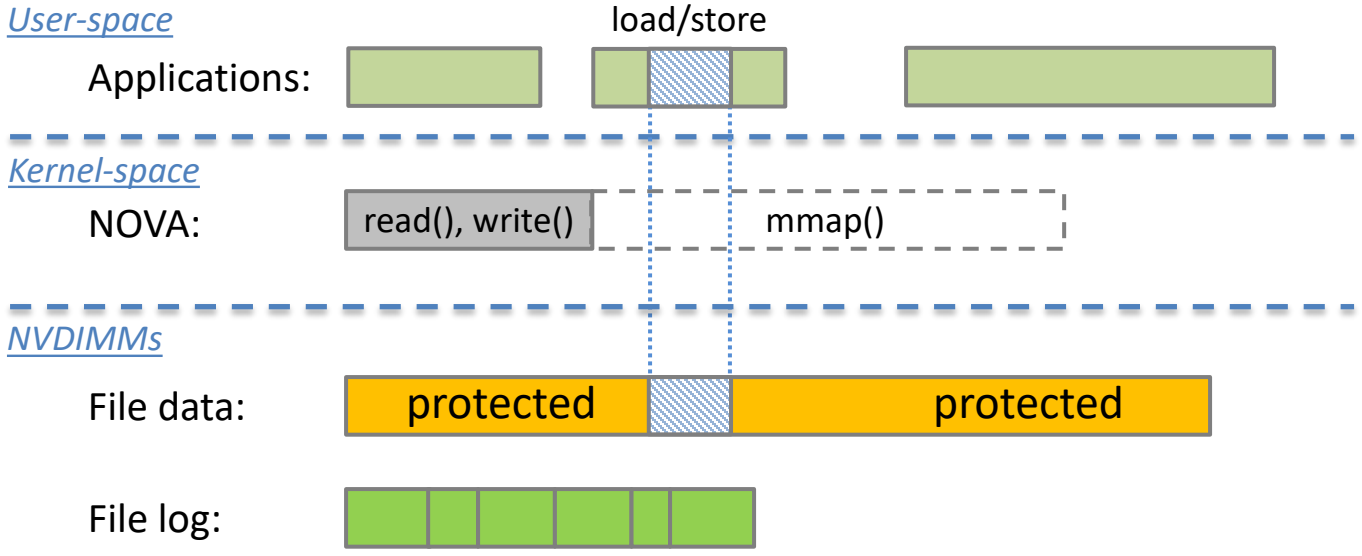


- If any step raises an error:
  - Attempt to repair and retry
  - If recovery fails: return `-EIO` to user

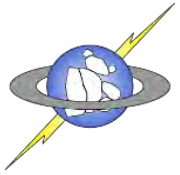


File data protection with DAX-mmap

- With DAX-Mmap(), file data changes are invisible to NOVA

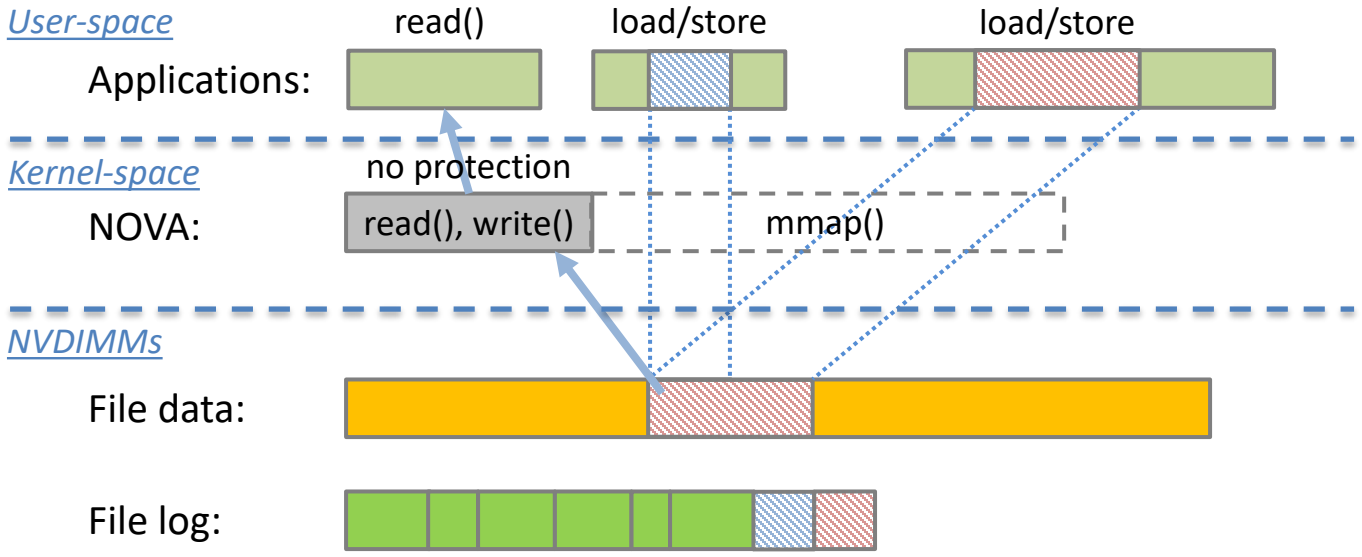


Following dax-mmap() semantics, NOVA doesn't interfere with mmap'ed file data.



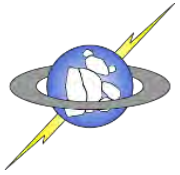
File data protection with DAX-mmap

- NOVA cannot protect mmap'ed file data
  - User applications directly load/store the mmap'ed region
  - NOVA has to know what file pages are mmap'ed



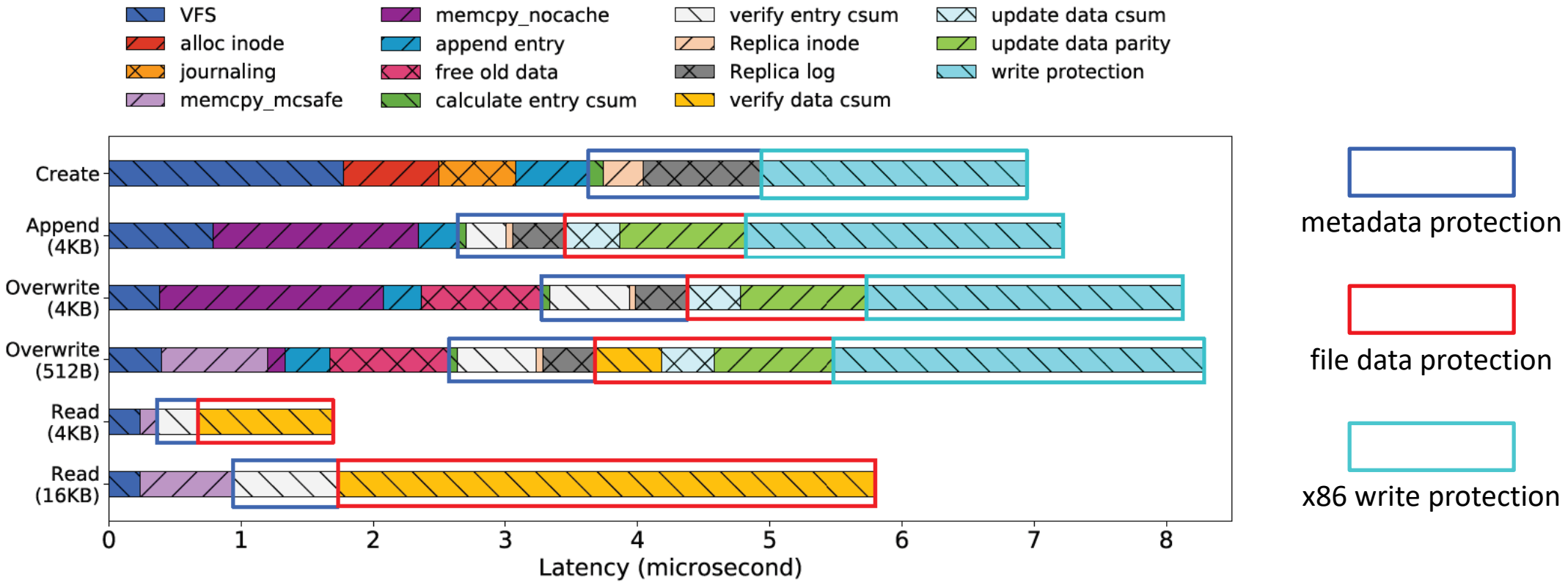
vm area list:

NOVA skips protection routines for reads and writes to the regions found in the vm area list.



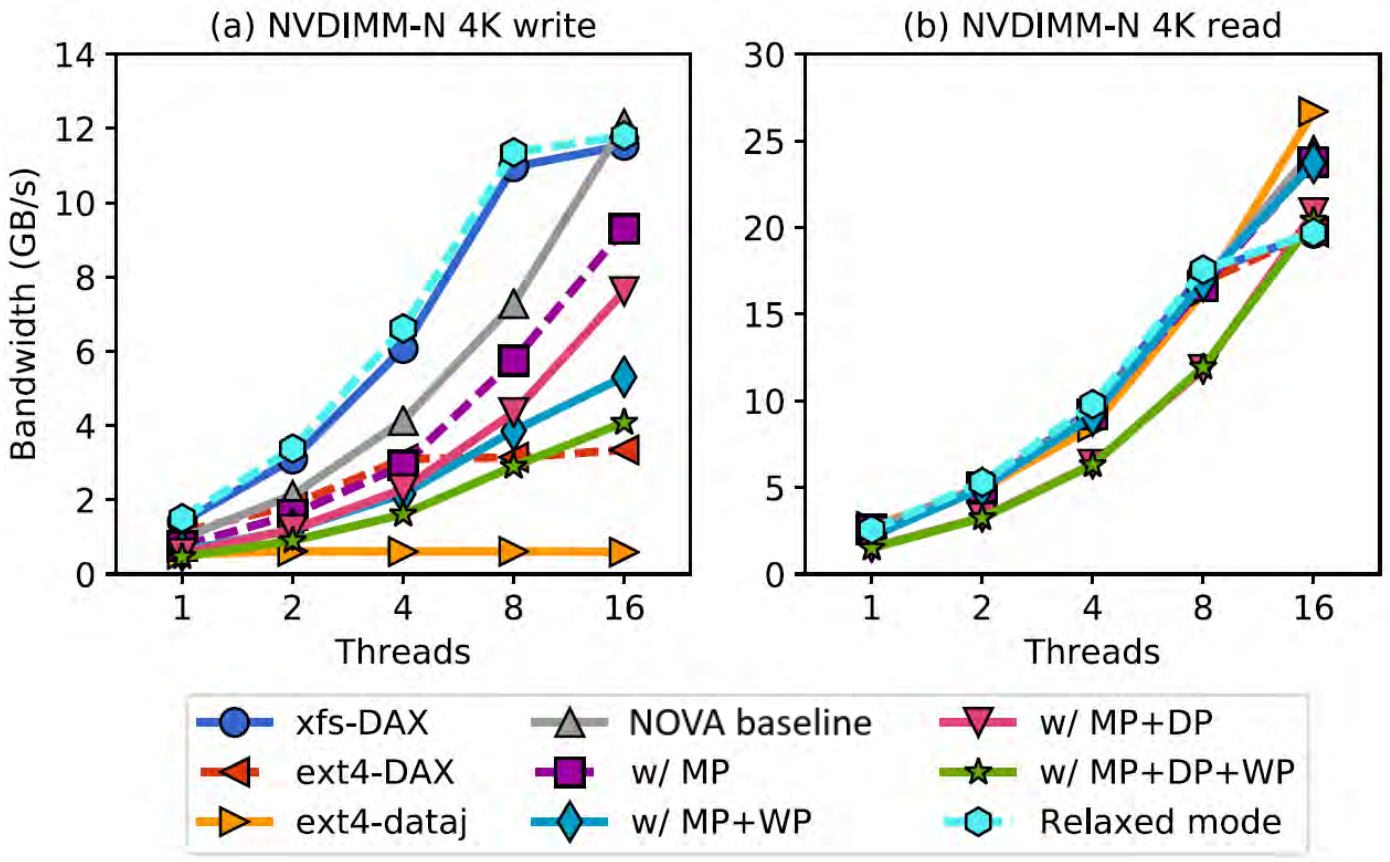
Performance impact of data integrity

- Latency breakdown on NVDIMM-N



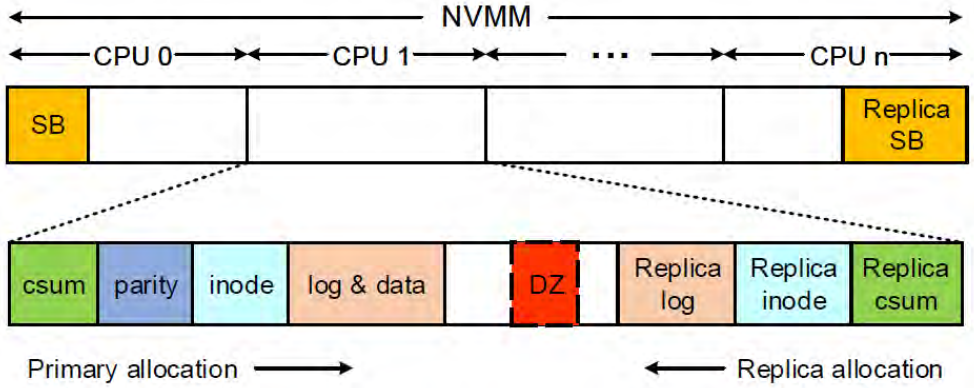
Performance impact of data integrity

- Random read/write bandwidth on NVDIMM-N



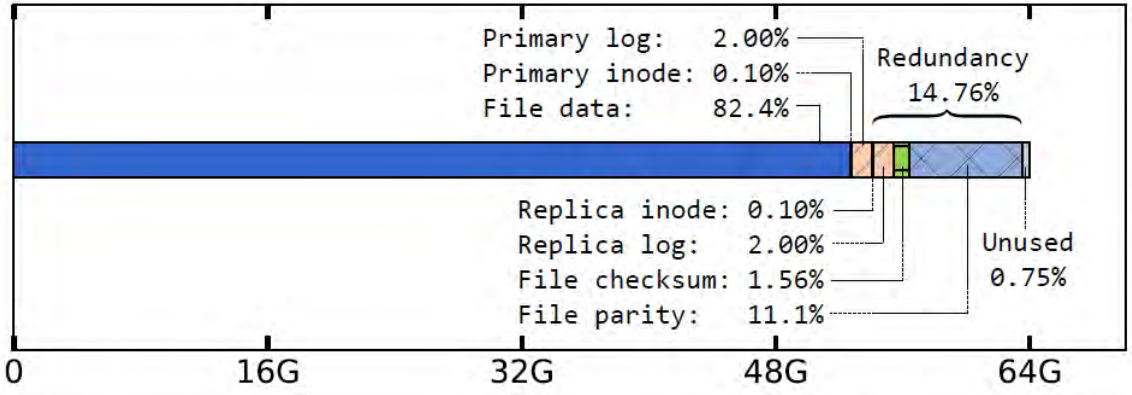
Storage utilization with data integrity

- Conceptual view of NOVA's utilization of NVMM



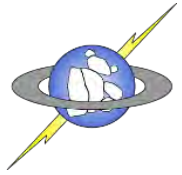
Dead-zone (DZ) only virtually exists to separate metadata replicas. File data can still live inside.

- Actual usage of a practical workload: fileserver



Differences from disk FS implementation

- Low-latency storage media
  - Need to choose fast methods for any involved computation
- Fine-grained random access
  - Need fine-grained checksum ranges, not per block (as in Btrfs, ZFS)
- Small atomicity guarantees (64-bit)
  - Need metadata replication to assist consistent updates
- Media errors cause machine check exceptions (MCEs)
  - Need awareness and mitigations
- Demands for DAX-mmap (no copy-on-write, no FS control)
  - Need awareness and lowering the protection level on demand

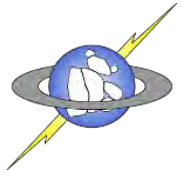


Recovery for snapshot metadata

- Snapshot metadata list resides in DRAM to reduce consistency overhead

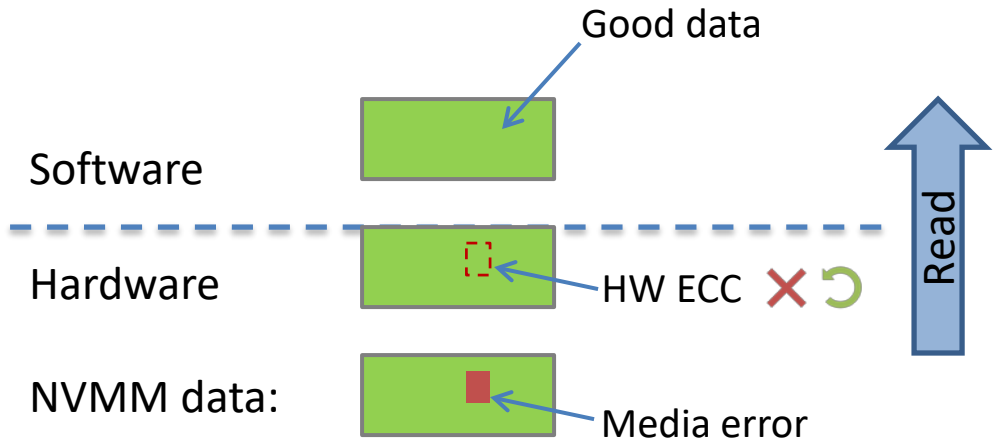
- Clean unmount:
  - Finish background snapshot delete
  - Save snapshot lists to NVMM

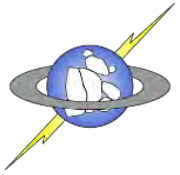
- Power failure:
  - Snapshot transaction ID is persistent
  - Rebuild snapshot metadata lists during power failure recovery



NVMM (Meta)data corruptions

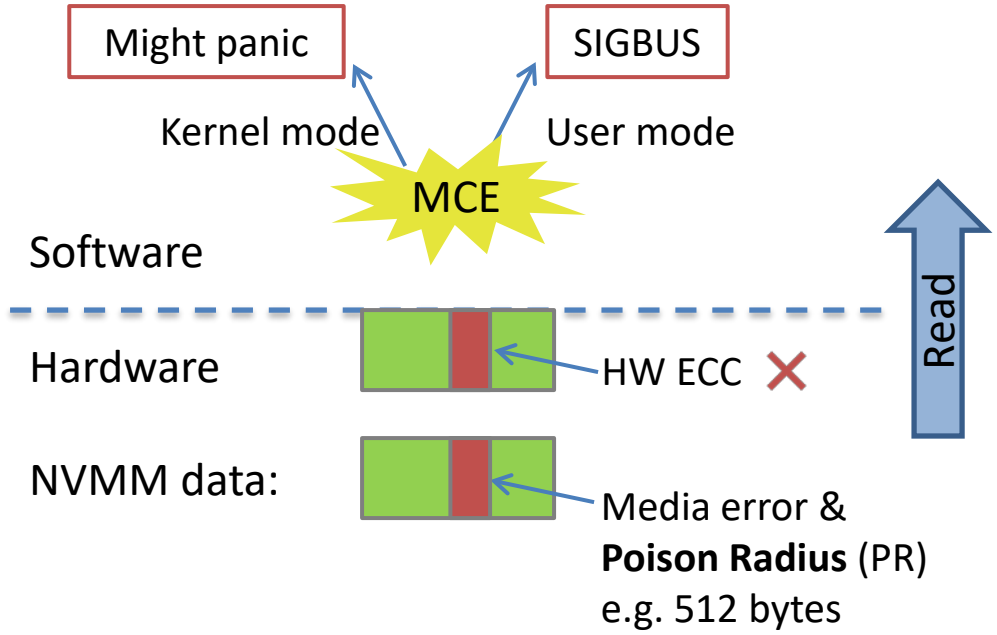
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - May consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

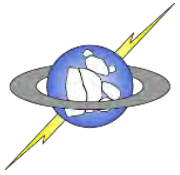




NVMM (Meta)data corruptions

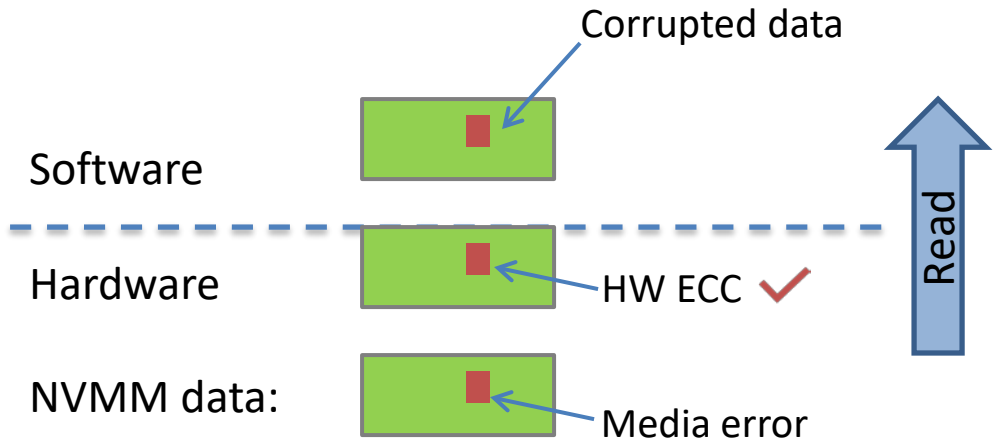
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - May consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

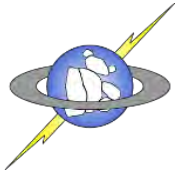




NVMM (Meta)data corruptions

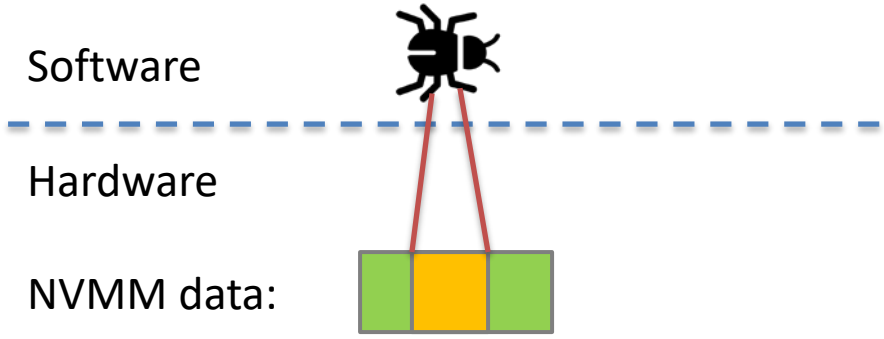
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

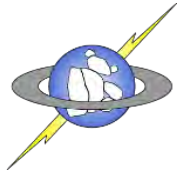




NVMM (Meta)data corruptions

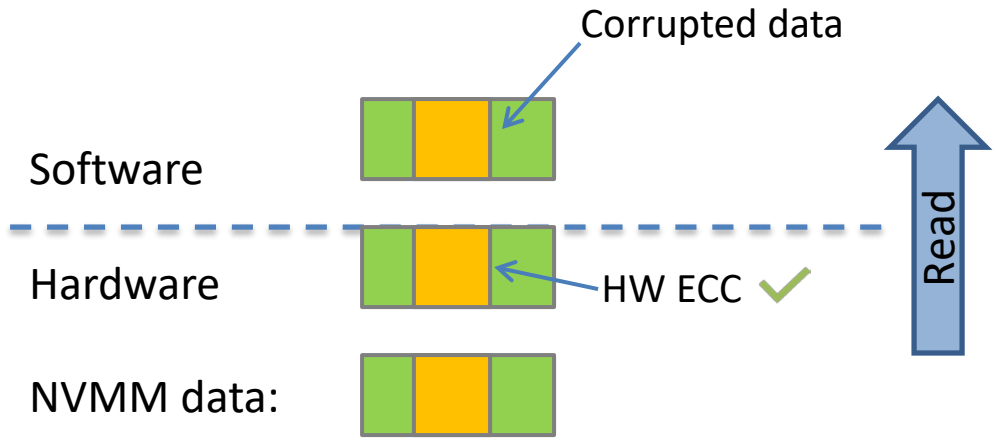
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware

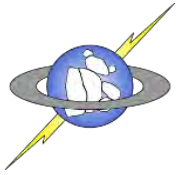




NVMM (Meta)data corruptions

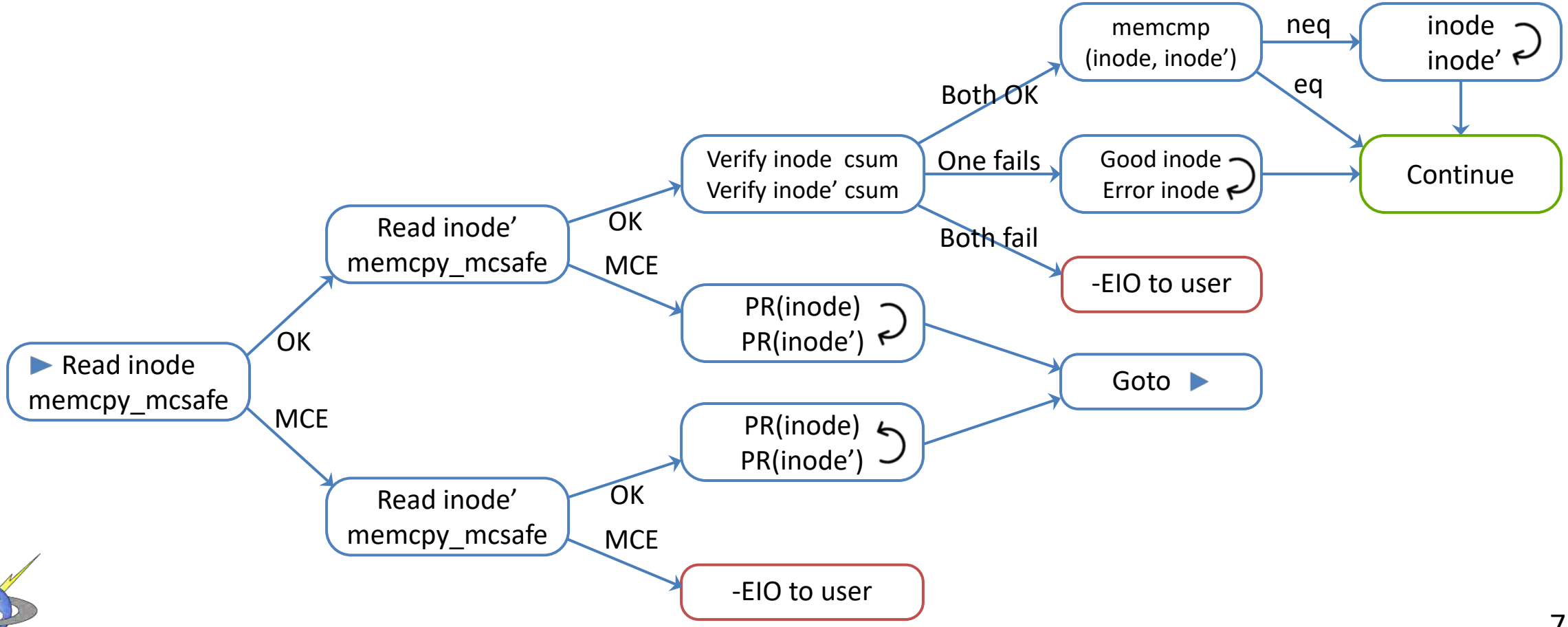
- Media errors
  - Detectable & correctable
    - Transparent to software
  - Detectable & uncorrectable
    - Affect a contiguous range of data
    - Raise machine check exception (MCE)
  - Undetectable
    - Consume corrupted data
- Software scribbles
  - Kernel bugs or own bugs
  - Transparent to hardware





Metadata error detection and correction

- Use inode access as example



File data error detection and correction

- Detect and repair both data and checksum errors