NOVA: A High-Performance, Fault Tolerant 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

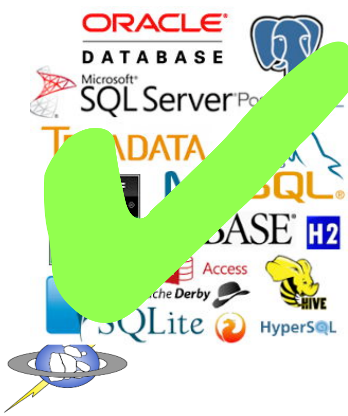
NVSL
Non-volatile Systems Laboratory

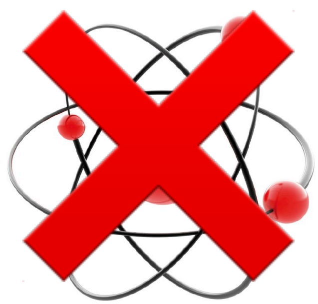
UCSDCSE
Computer Science and Engineering

NVMM File System Requirements

- Legacy File IO Acceleration – fast and easy
  - Run existing IO-intensive apps on NVDIMMs
  - "just works"
- Strong atomicity
- DAX Mmap
  - Load-store access & complex data structures
  - More control/responsibility for programs
- Data protection
  - Don't trust the media or other software
  - Support backups
- High performance
  - Otherwise, why bother?

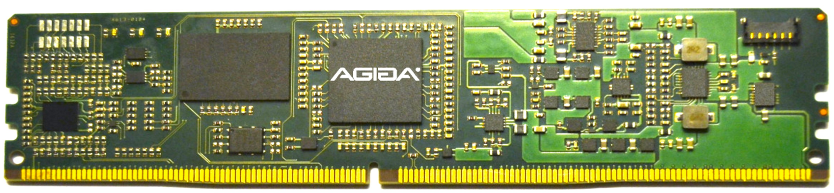
XFS EXT4 F2FS BTRFS NILFS

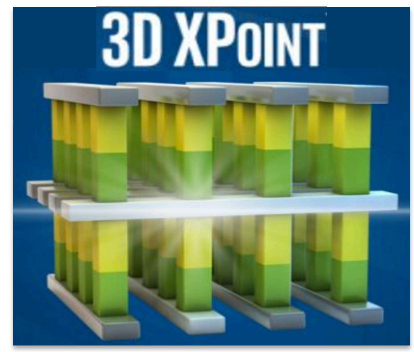


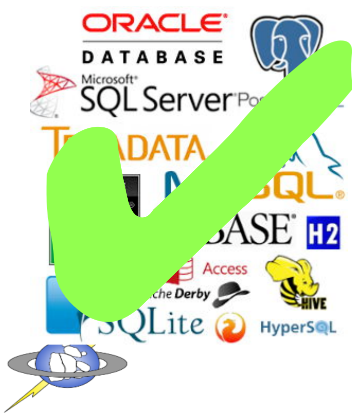


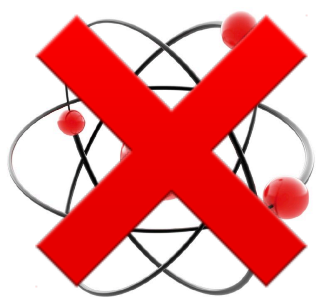
BPFS SCMFS PMFS Aerie

EXT4-DAX XFS-DAX M1FS

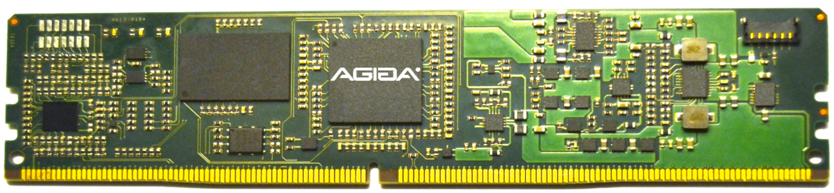


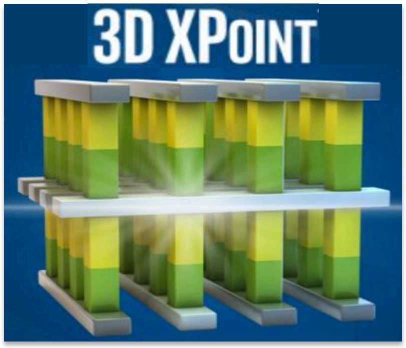


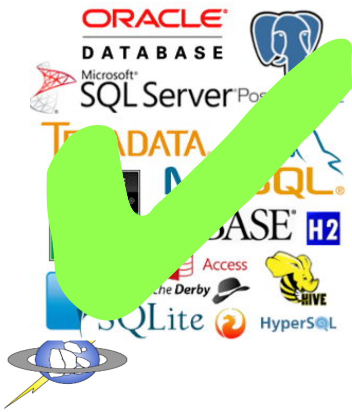


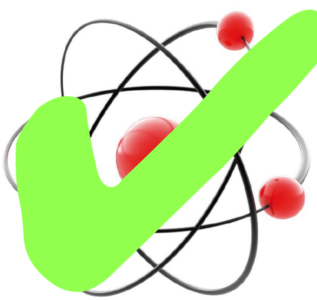


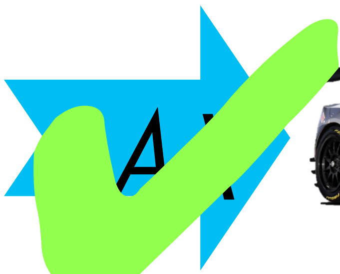
NOVA



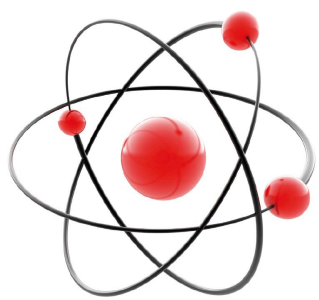

3D XPoint


ORACLE
DATABASE
Microsoft
SQL Server
H2
Access
HIVE
HyperSQL



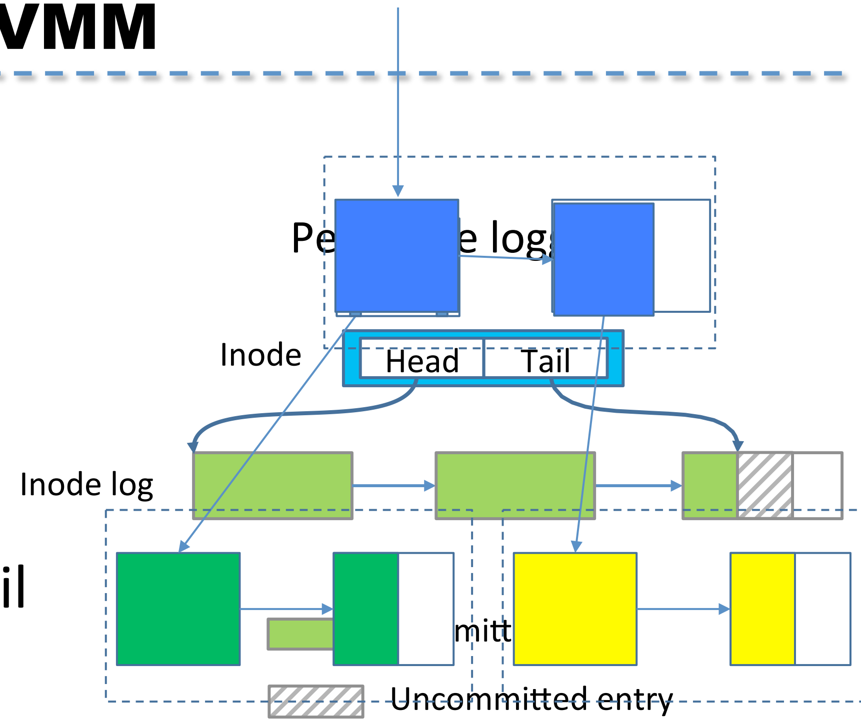


Files, Atomicity, and Performance



Log Structured FS For NVMM

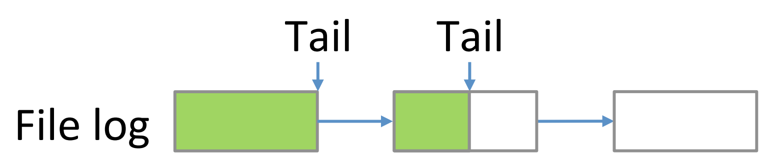
- A Nova FS is a tree of logs
  - One log per i-node
  - Logs are not contiguous
- I-nodes point to the head and tail



Atomicity: Logging for Simple Metadata Operations

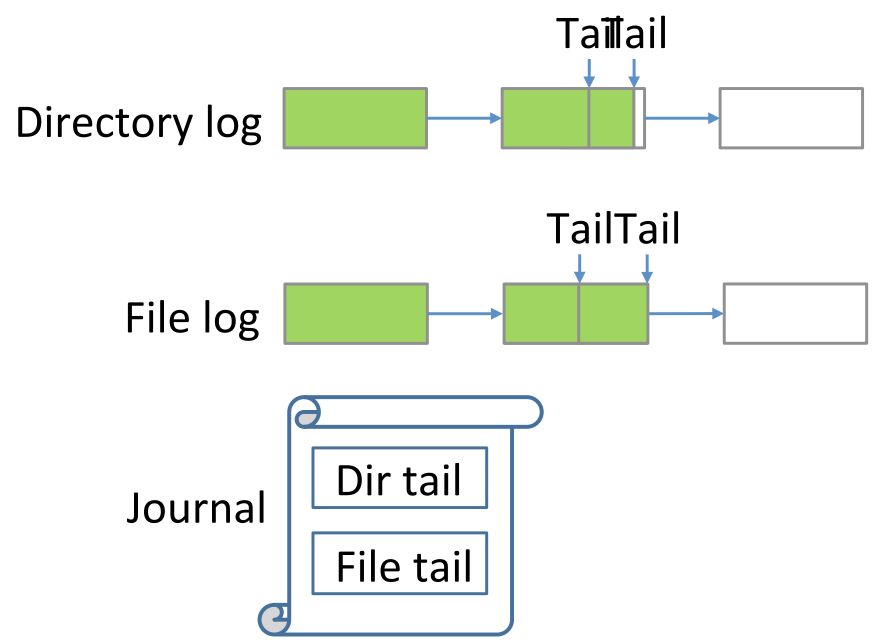
- Combines log-structuring, journaling and copy-on-write

- Log-structuring for single log update
  - Write, msync, chmod, etc
  - Lower overhead than journaling and shadow paging



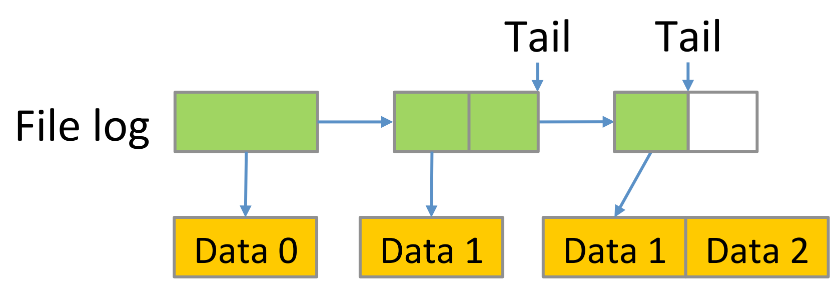
Atomicity: Lightweight Journaling for Complex Metadata Operations

- Lightweight journaling for update across logs
  - Unlink, rename, etc
  - Journal log tails instead of metadata or data



Atomicity: Copy-on-write for file data

- Copy-on-write for file data
  - Log only contains metadata
  - Log is short

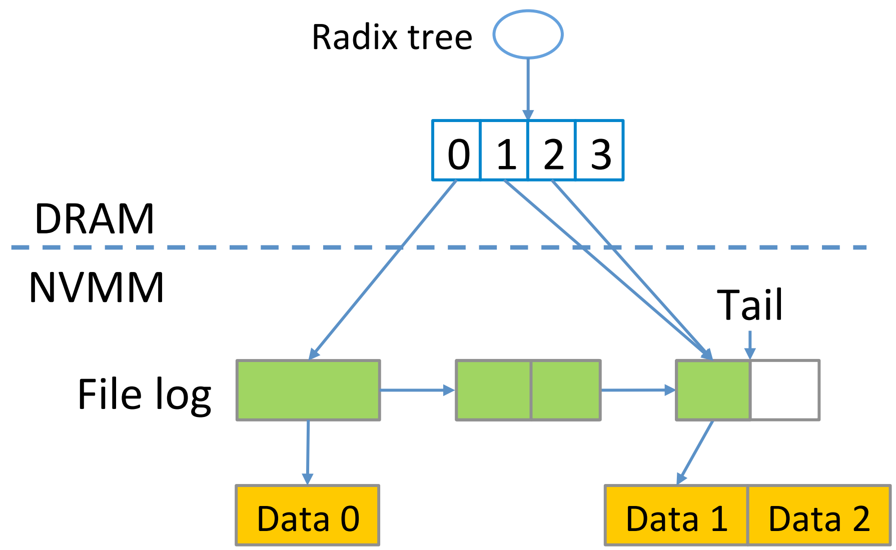


Atomicity: DAX

- Nova does not make atomicity guarantees for DAX mmap()'d data
- msync() works, but it's slow & non-atomic
- The program must ensure consistency
  - ISA Support: CLWB, clflush, etc.
  - Careful programming

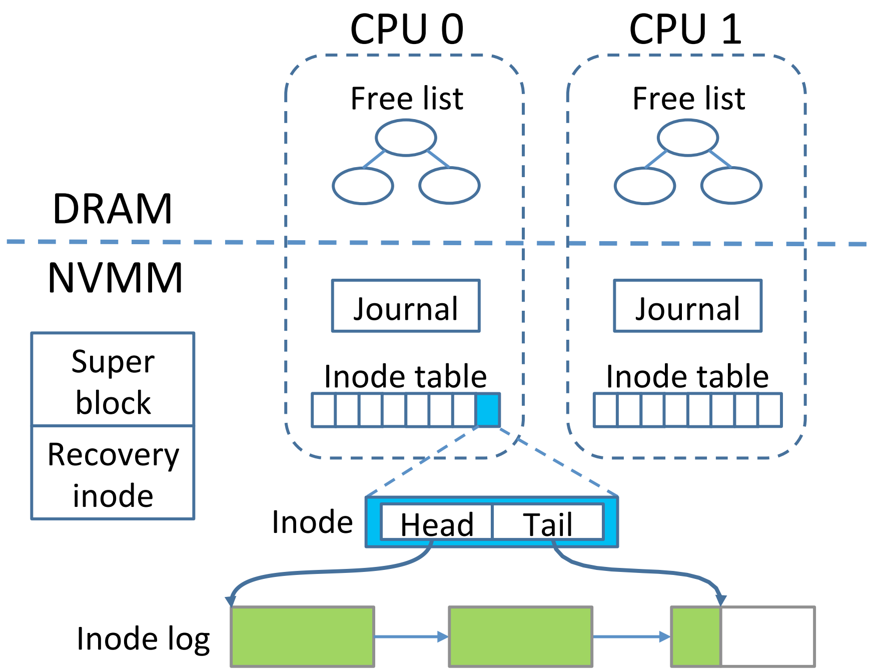
DRAM Indexes for High Performance

- 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



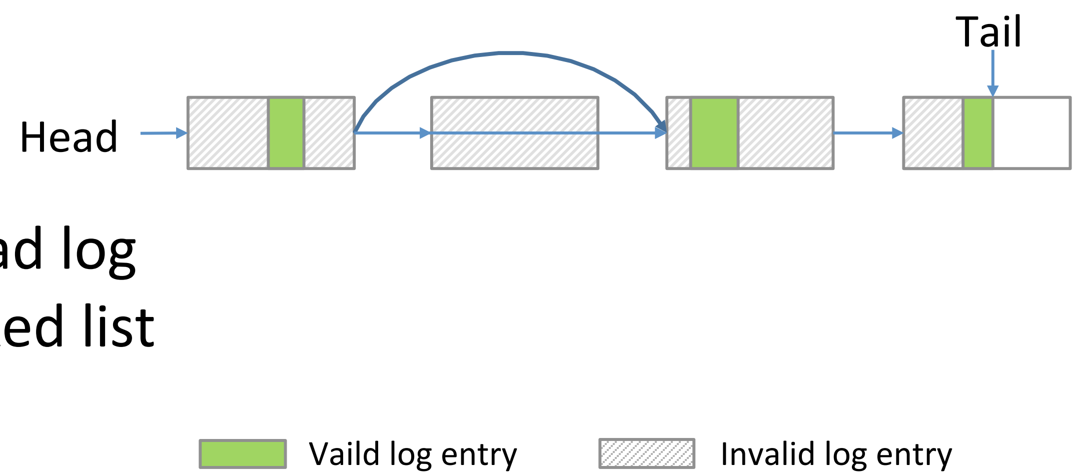
Performance and Scalability

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



Fast garbage collection

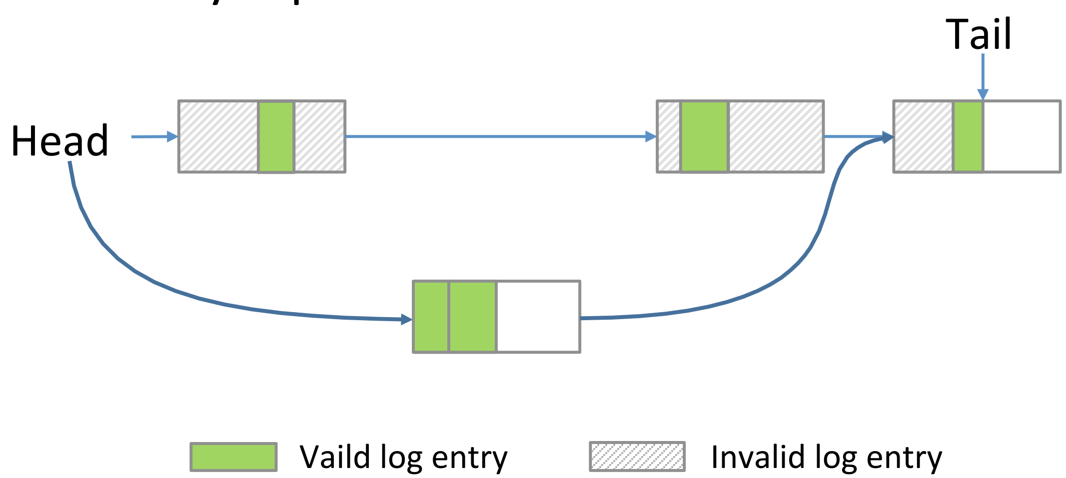
- Log is a linked list
- Log only contains metadata



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

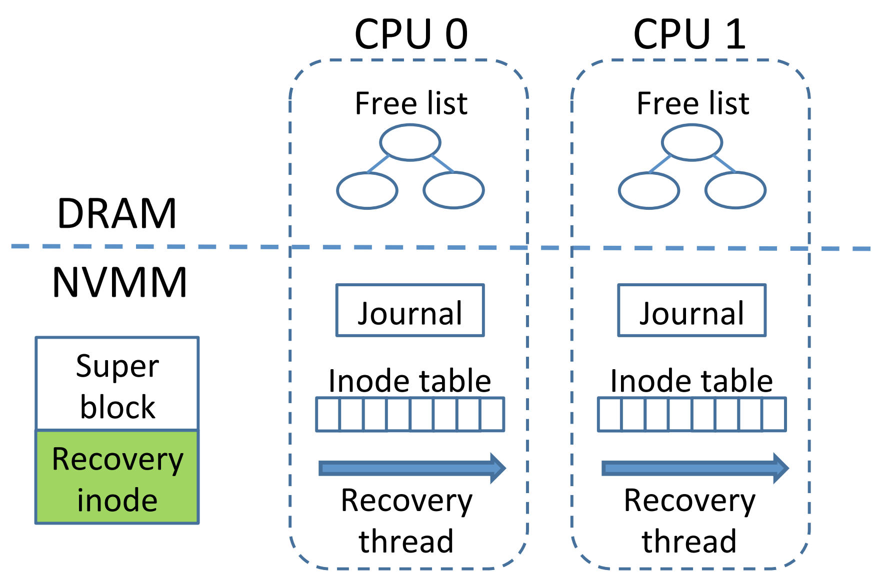
Thorough garbage collection

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

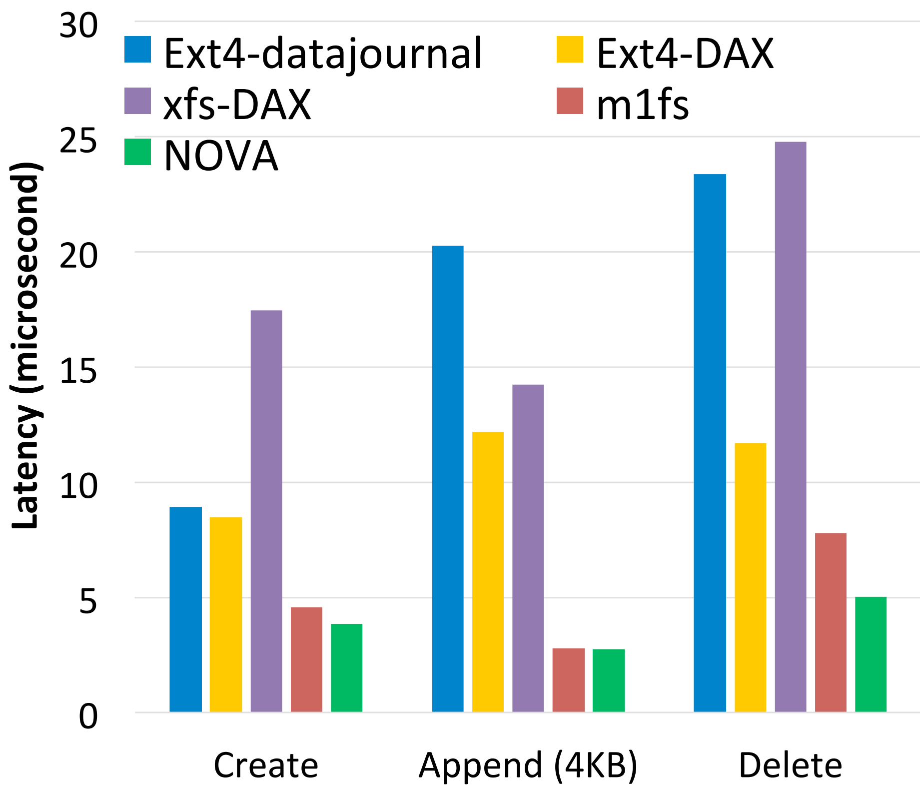


Recovery

- Normal shutdown recovery:
  - Store allocator state in recovery inode
  - Constant time startup
- Failure recovery:
  - Parallel scan
  - Failure recovery bandwidth: > 400 GB/s

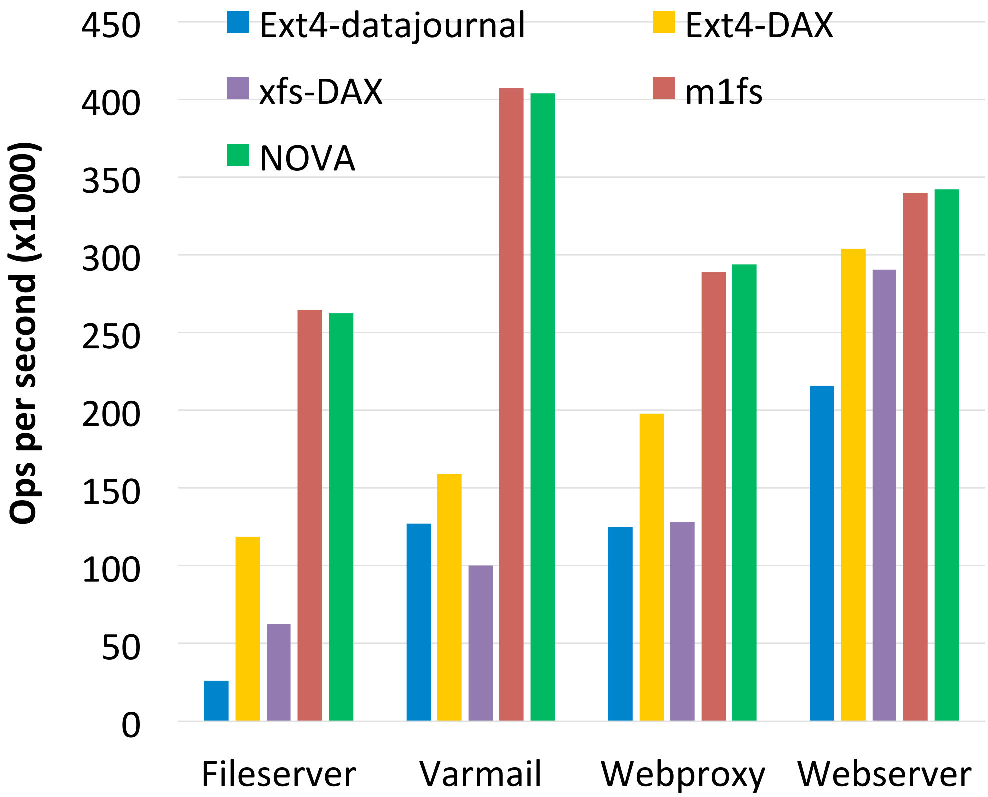


Operation Latency



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

Filebench throughput



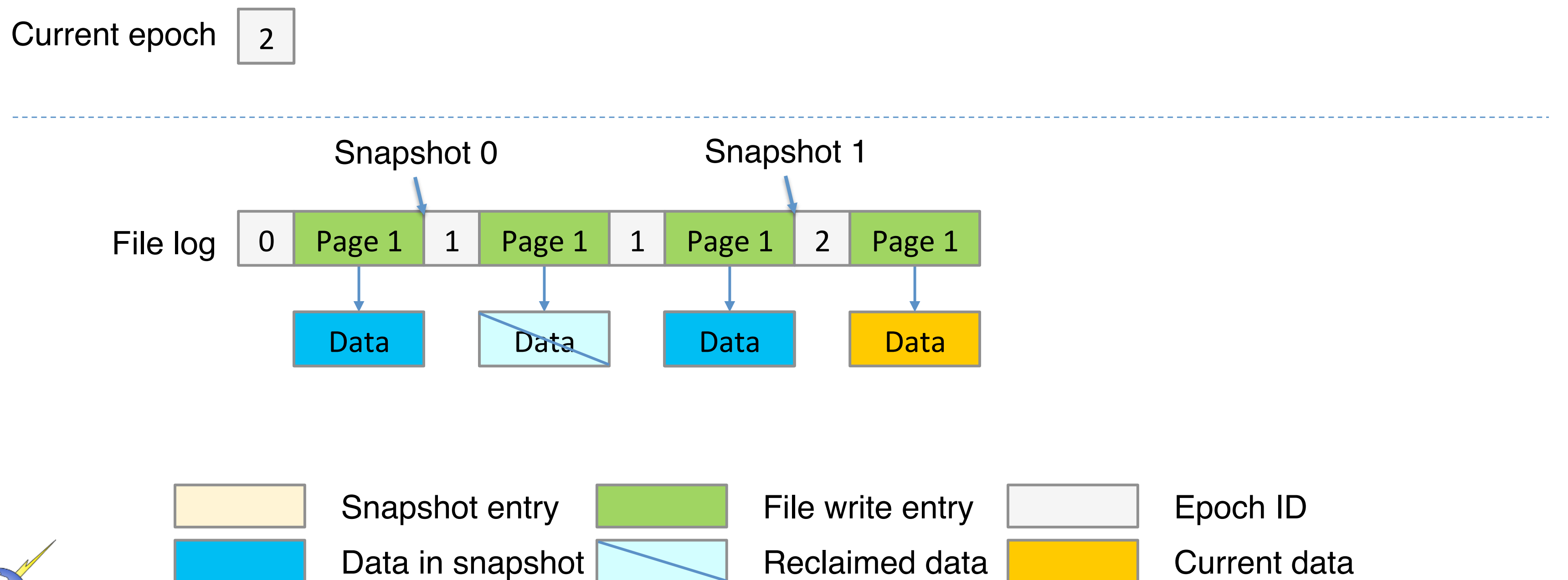
- NOVA achieves high performance with strong data consistency

DAX, Backup, and Data Protection

DAX Challenges

- DAX mmap() gives the programmer great power
  - Fine-grain updates
  - Pointer-based data structures
  - Custom data layout
- …along with great responsibilities.
  - Data structure consistency/persistence ordering
  - Memory protection/media error management
- The file system must not interfere.

Snapshots for Normal File Access



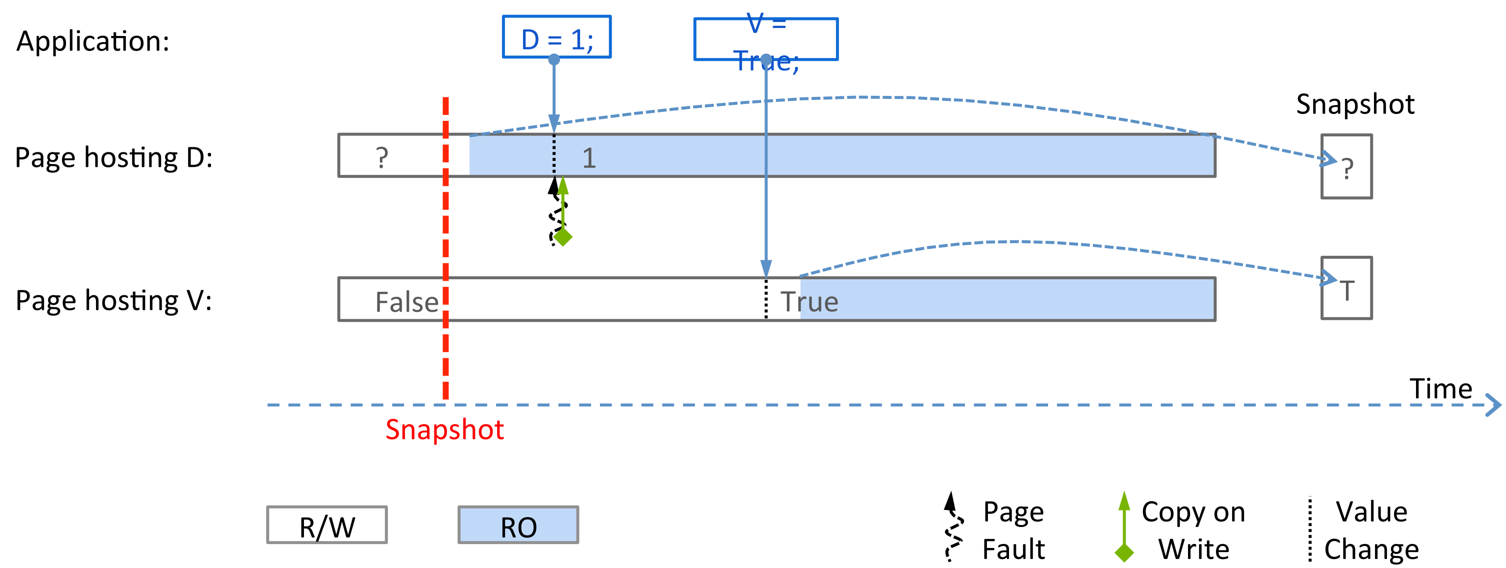
Memory Ordering With DAX mmap()

```
D = 42;
V = true;
```

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

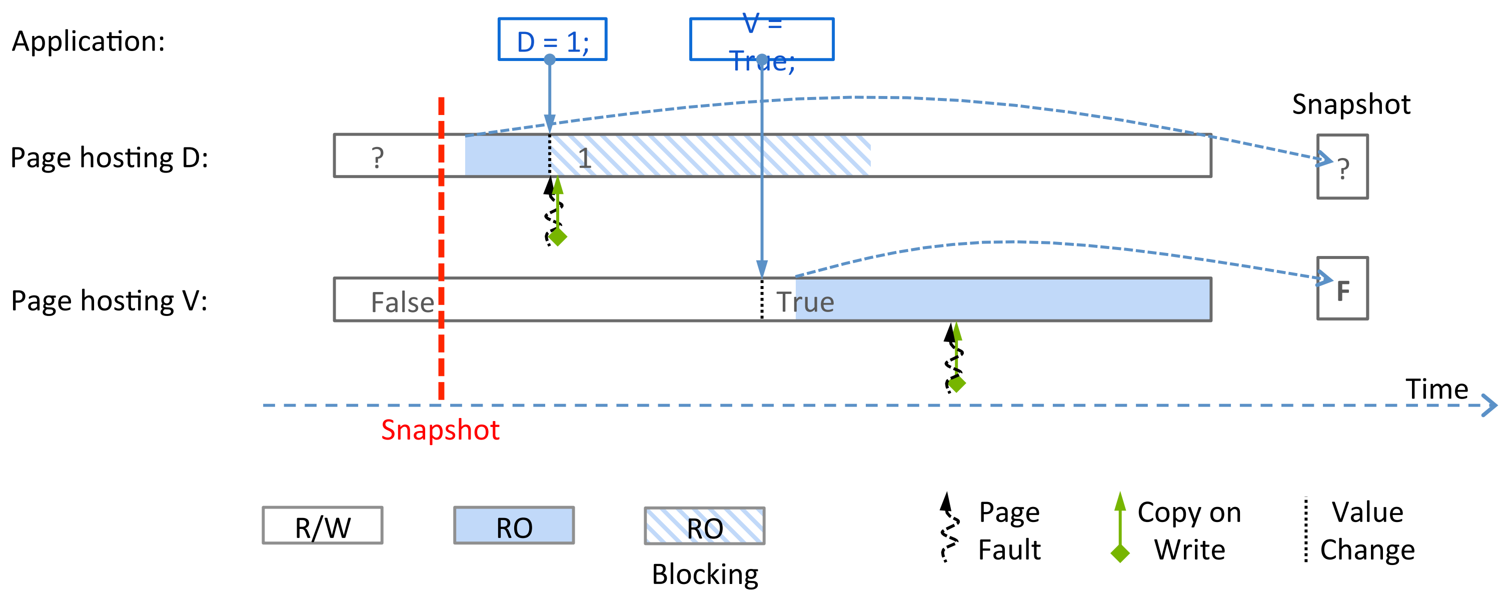
Corrupt Snapshots with DAX-mmap()

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



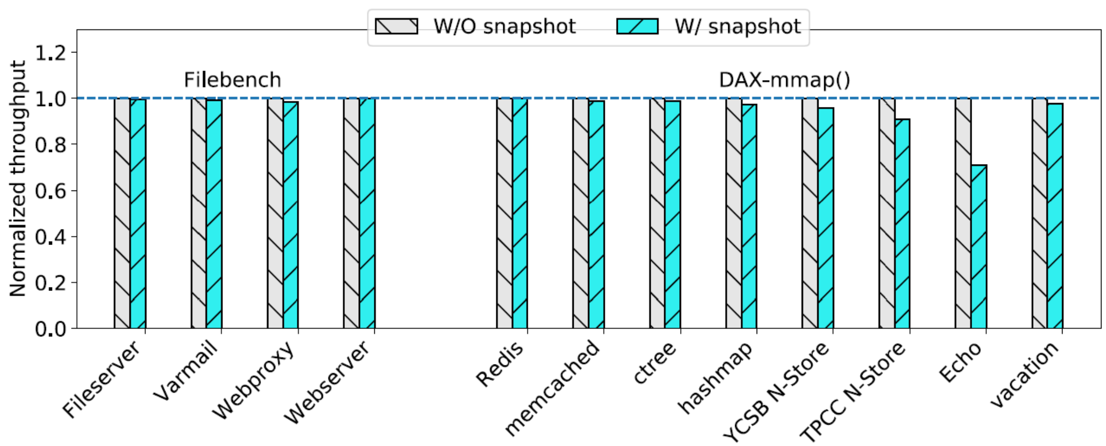
Consistent Snapshots with DAX-mmap()

- 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()

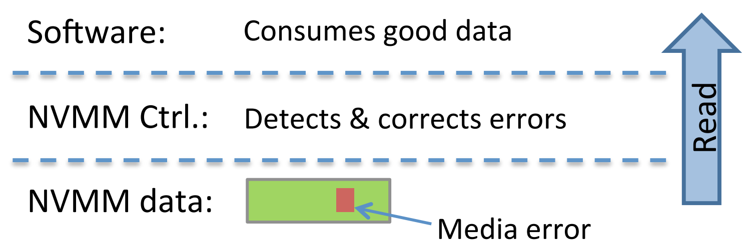


Conventional workloads NVMM-aware workloads from WHISPER

Protecting Metadata

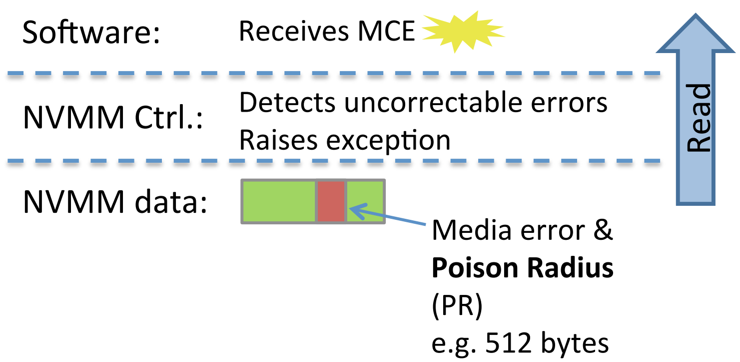
NVMM Failure Modes: Media Failures

- 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



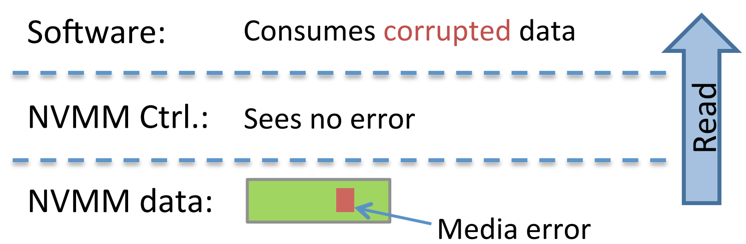
NVMM Failure Modes : Media Failures

- 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



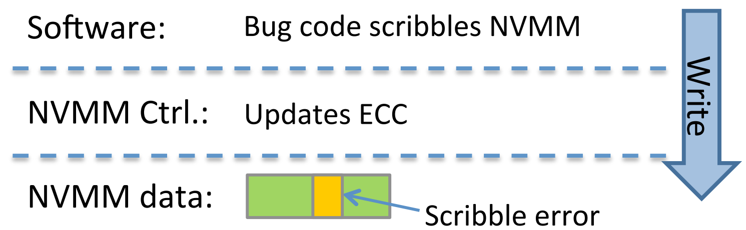
NVMM Failure Modes : Media Failures

- 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



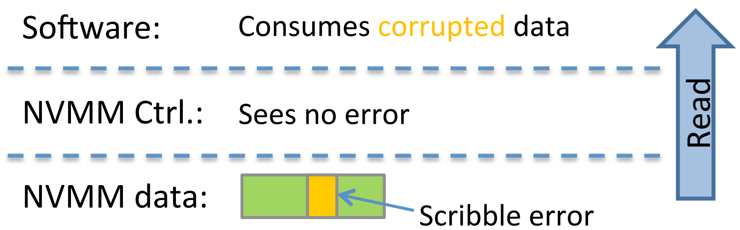
NVMM Failure Modes: Scribbles

- 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



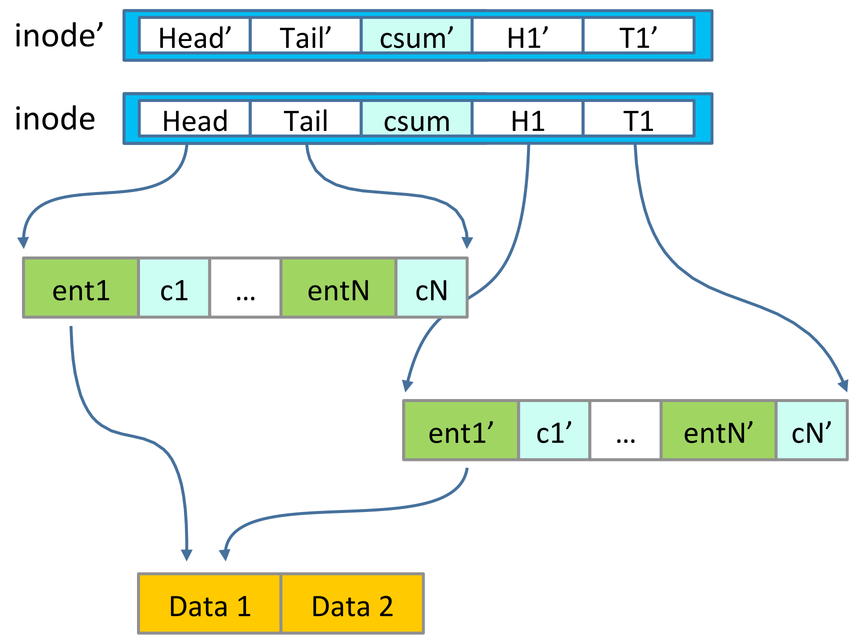
NVMM Failure Modes: Scribbles

- 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

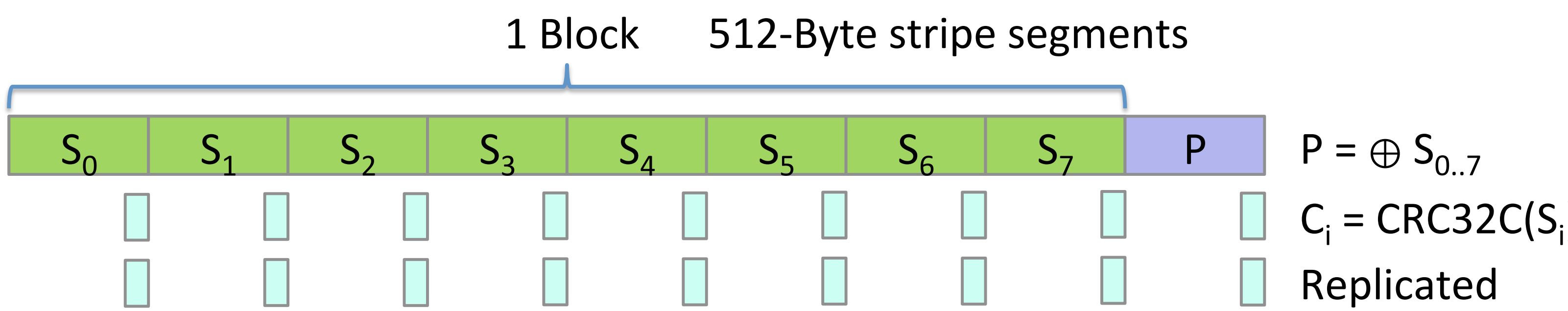
- Tolerating Larger Scribbles
  - Allocate replicas far from one another
  - Can tolerate arbitrarily large scribbles to metadata.
- Preventing scribbles
  - Mark all NVMM as read-only
  - Disable CPU write protection while accessing NVMM

Protecting Data

DAX

NOVA Data Protection

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



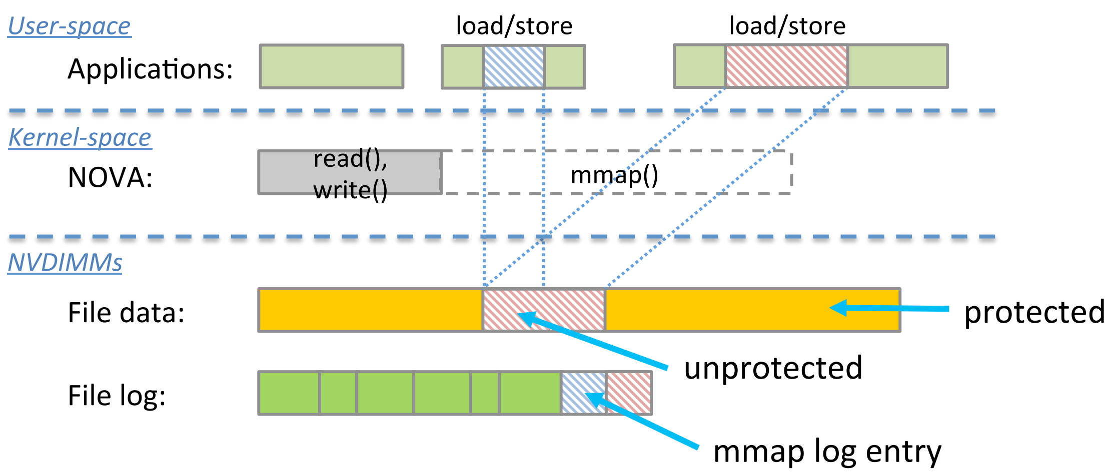
Data Protection With DAX mmap()

- File systems cannot efficiently protect mmap'ed data, since stores are invisible
- NOVA's data protection contract:

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

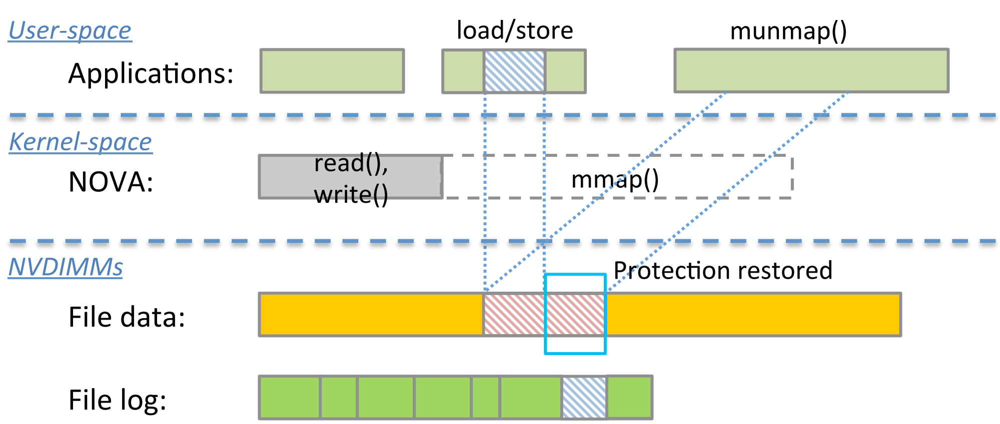
File data protection with DAX-mmap

- NOVA logs mmap() operations



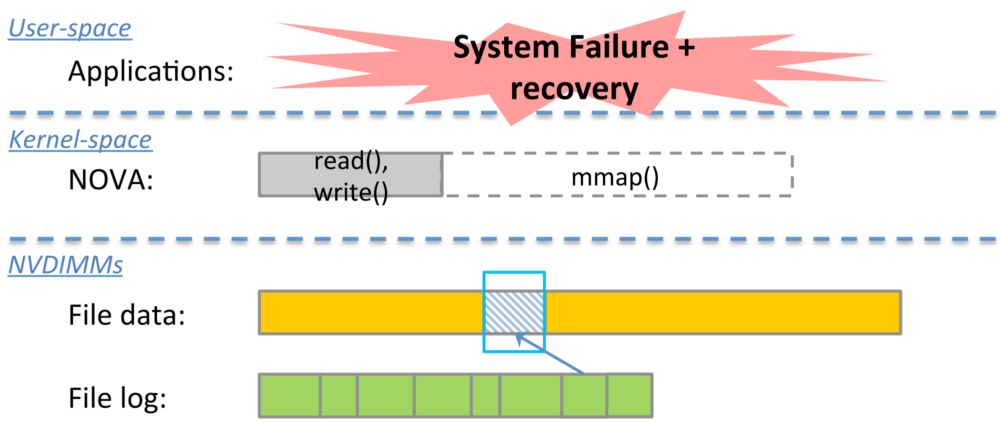
File data protection with DAX-mmap

- On unmap and during recovery, NOVA restores protection

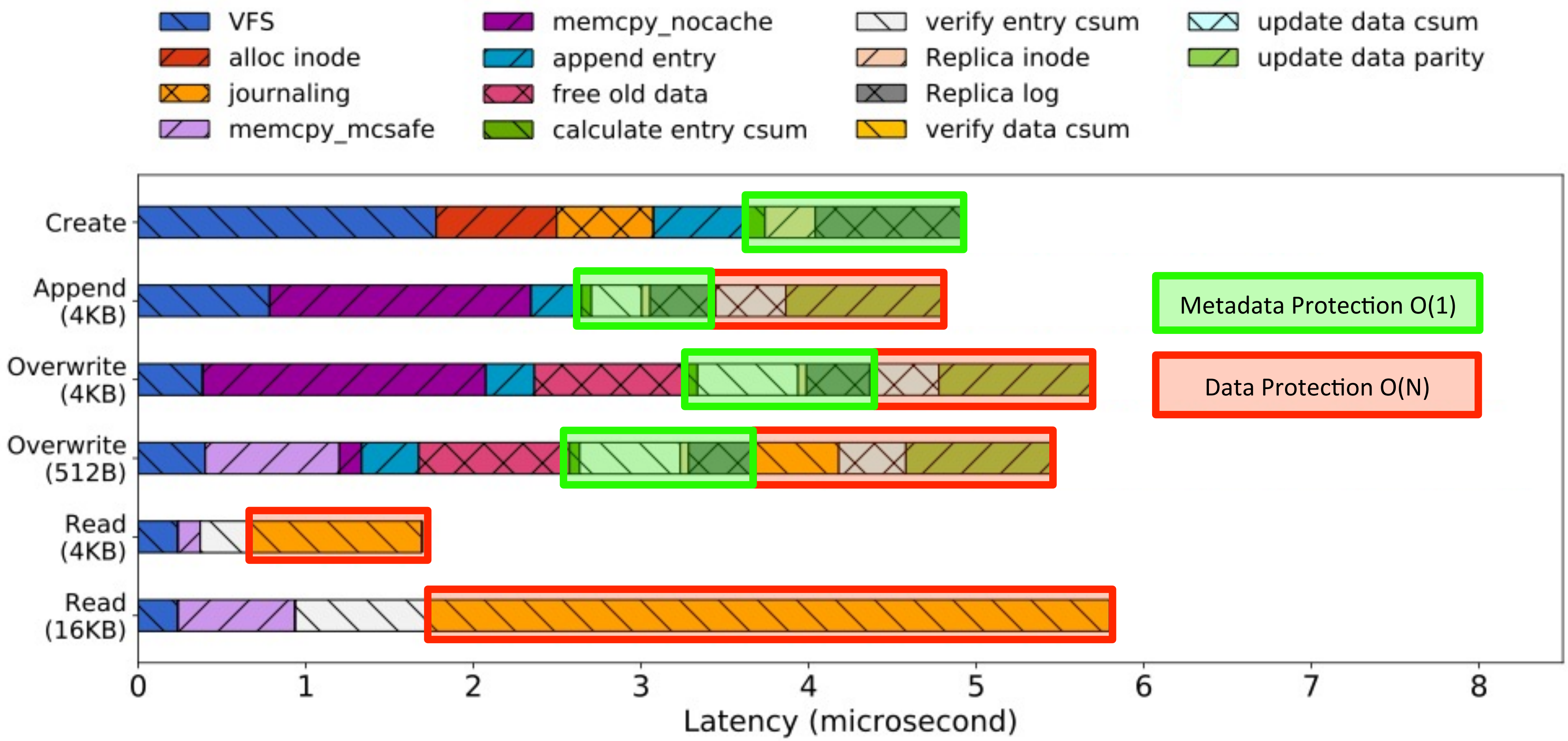


File data protection with DAX-mmap

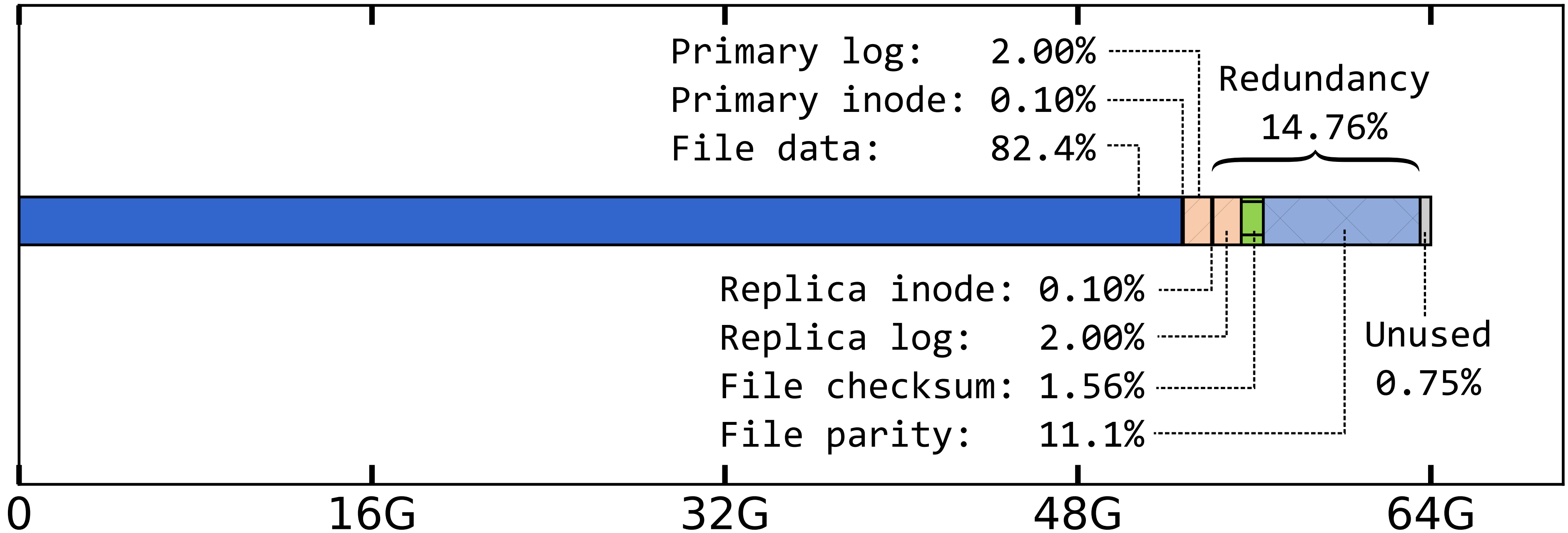
- On unmap and during recovery, NOVA restores protection



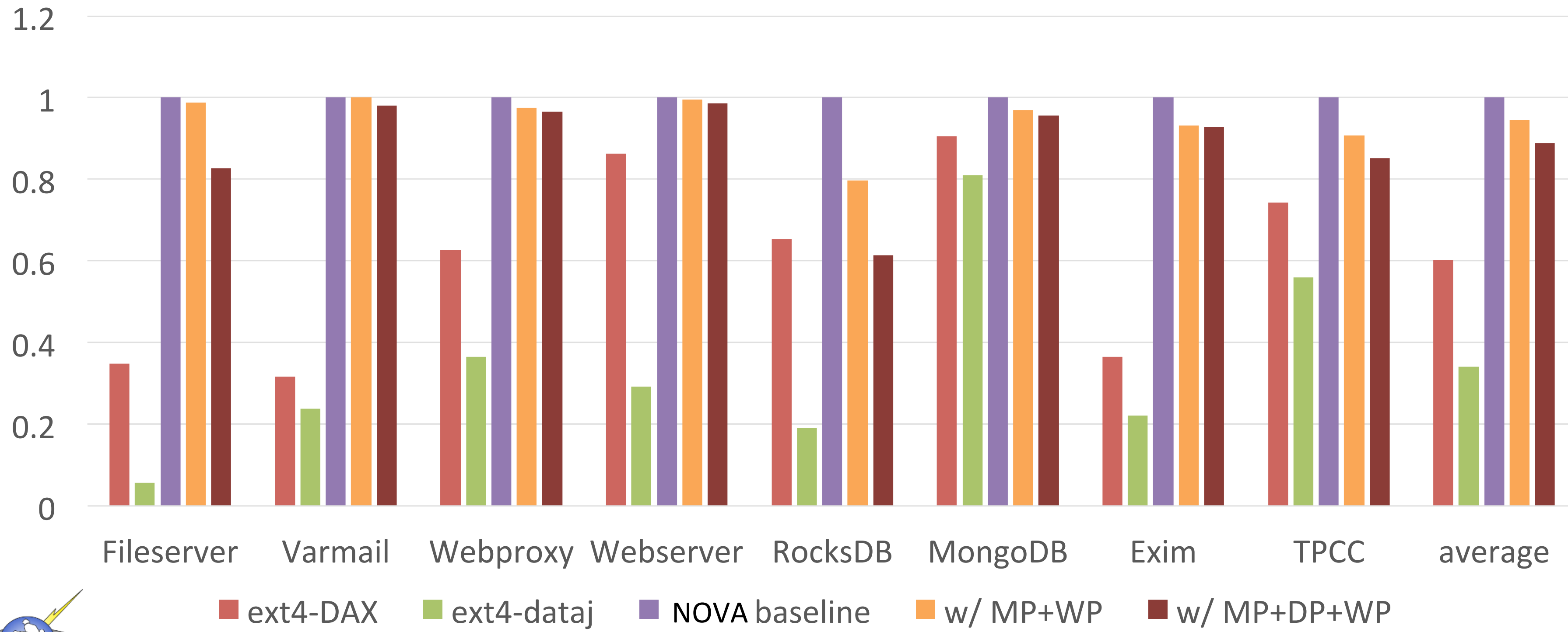
Performance


VFS
alloc inode
journaling
memcpy_mcsafe
memcpy_nocache
append entry
free old data
calculate entry csum
verify entry csum
Replica inode
Replica log
verify data csum
update data csum
update data parity
Create
Append (4KB)
Overwrite (4KB)
Overwrite (512B)
Read (4KB)
Read (16KB)
Metadata Protection O(1)
Data Protection O(N)
0
1
2
3
4
5
6
7
8
Latency (microsecond)

Storage Utilization



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

NOVA is open source. We are preparing it for addition to Linux.

To help or try it out: https://github.com/NVSL/linux-nova

Thanks!