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

Jian Xu* Lu Zhang* Amirsaman Memaripour Akshatha Gangadharaiah
Amit Borase Tamires Brito Da Silva Steven Swanson Andy Rudoff+
University of California, San Diego Intel Corporation+

## ABSTRACT

Emerging fast, persistent memories will enable systems that combine conventional DRAM with large amounts of non-volatile main memory (NVMM) and provide huge increases in storage performance. Fully realizing this potential requires fundamental changes in how system software manages, protects, and provides access to data that resides in NVMM. We address these needs by describing an NVMM-optimized file system called NOVA-Fortis that is both fast and resilient in the face of corruption due to media errors and software bugs. We identify and propose solutions for the unique challenges in adding fault tolerance to an NVMM file system, adapt state-of-the-art reliability techniques to an NVMM file system, and quantify the performance and storage overheads of these techniques.

## 1 INTRODUCTION

Integrating fast NVMM technologies (e.g., battery-backed NVDIMMs or Intel and Micron's 3D XPoint) into computer systems presents a host of interesting challenges on how we should redesign existing software components (e.g., file systems) to accommodate and exploit the unique characteristics that NVMMs provide. Several groups have proposed new file systems [1–3] designed specifically for NVMMs. Windows (NTFS) and Linux file systems (Ext4-DAX and XFS-DAX) now also include at least rudimentary support for them.

Despite these NVMM-centric performance improvements, none of these file systems provide the data protection features necessary to detect and correct media errors, protect against data corruption due to misbehaving code, or perform consistent backups of the NVMM's contents. File system stacks in wide use (e.g., ext4 running atop LVM, Btrfs, and ZFS) provide some or all of these capabilities for block-based storage. If users are to trust NVMM file systems with critical data, they will need these features as well.

From a reliability perspective, there are four key differences between conventional block-based file systems and NVMM file systems.

*The first two authors contributed equally to this work. It is accepted to the 26th ACM Symposium on Operating Systems Principles (SOSP'17). http://cseweb.ucsd.edu/~swanson/papers/SOSP2017-NOVAFortis.pdf

First, the memory controller reports NVMM media errors as non-maskable interrupts rather than error codes from a block driver.

Second, NVMM file systems must support DAX-style memory mapping that maps NVMM pages directly into the application's address space. However, it means a file's contents can change without the file system's knowledge.

Third, the entire file system resides in the kernel's address space, vastly increasing vulnerability to "scribbles" – errant stores from misbehaving kernel code.

Fourth, NVMMs are vastly faster than block-based storage devices. This means that the trade-offs block-based file systems make between reliability and performance need a thorough re-evaluation.

We explore the impact of these differences on file system reliability by building *NOVA-Fortis* [4], an NVMM file system that adds fault-tolerance to NOVA [3] by incorporating snapshots, replication, checksums, and RAID-4 parity.

## 2 NOVA-FORTIS FILE SYSTEM

NOVA-Fortis extends NOVA, a log-structured file system for systems that include NVMM along with DRAM. Each NOVA's inode maintains pointers to a per-inode log that comprises of a linked list of 4 KB log pages, where a head pointer saves the start address of the log and a tail pointer always points to the latest committed log entry. NOVA uses processor's 64-bit atomic writes to commit updates atomically to the log by updating the inode's log tail pointer.

NOVA stores file data in 4 kB pages and uses copy-on-write when updating existing data pages. The log entry for a write operation holds pointers to the newly written pages, atomically replacing them in the file. NOVA immediately reclaims the resulting stale pages.

### 2.1 NOVA-Fortis Snapshots

Snapshots provide a consistent image of the file system at a moment in time. They facilitate consistent backups without unmounting the file system, affording protection against system failures and the accidental deletion or modification of files. Neither existing DAX-enabled NVMM file systems nor current low-level NVMM drivers support snapshots, making consistent online backups impossible.

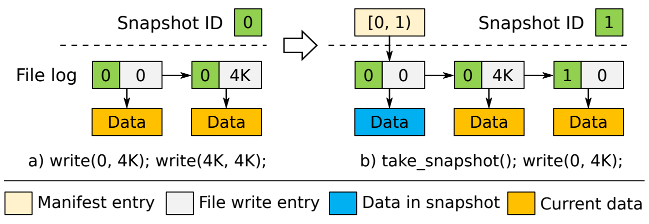

**Figure 1: NOVA-Fortis snapshot – File pages 0~4K and 4K~8K are in snapshot 0. Then file page 0~4K is overwritten in the current state.**

NOVA-Fortis implements snapshots by maintaining a global snapshot ID for the file system and storing the current snapshot ID in each log entry. Creating a new snapshot increments the global snapshot ID, and it does not block file system write operations. To open files of a snapshot, NOVA-Fortis traverses the logs only while the log entries' snapshot IDs are smaller than or equal to the target snapshot ID.

When writing to a file, NOVA-Fortis does not immediately reclaim file pages belonging to an earlier snapshot. Instead, NOVA-Fortis maintains a *snapshot manifest* for each snapshot to reclaim stale pages. Entries in the manifest contain a pointer to a log entry, and a pair of lifespan snapshot IDs denoted by [*create ID, delete ID*) as shown in Figure 1.

Taking consistent snapshots while applications are modifying files using DAX-style `mmap` requires NOVA-Fortis to mark each of the read/write mapped pages as read-only and then do copy-on-write when a store occurs to preserve the old pages as part of the snapshot. However, a naive approach can leave the snapshot in an inconsistent state when the order of setting pages read-only is incompatible with the ordering that the application requires.

NOVA-Fortis resolves the challenge by modifying the kernel's page fault handler to handover to NOVA-Fortis where it prevents the page fault to read-only pages from completing until it can finish creating the snapshot.

### 2.2 NOVA-Fortis Data Integrity

Like all storage media, NVMM is subject to data and metadata corruption from media failures and software bugs. To prevent, detect, and recover from data corruption, NOVA-Fortis relies on the capabilities of the system hardware and operating system as well as its own mechanisms.

NOVA-Fortis assumes the NVMM subsystem provides hardware-based ECC to transparently correct correctable errors, and silently return invalid data for undetectable ones.

For detectable but uncorrectable NVMM errors an Intel platform raises an *Machine Check Exception (MCE)*, and the default response to an MCE in the Linux kernel is a panic. However, recent Linux kernels include a version of `memcpy()`, called `memcpy_mcsafe()`, that returns an error to the caller instead of crashing in response to memory-error-induced MCEs. NOVA-Fortis always uses this function when reading from NVMM and checks its return code to detect hardware-uncorrectable media errors.

To be able to recover from any errors NOVA-Fortis protects its metadata by keeping two copies of each structure – a *primary* and a *replica* – and adding a CRC32 checksum to both, in order to detect software-induced errors.

NOVA-Fortis adopts RAID-4 parity protection and checksums to protect file data. It treats each 4 KB file page as a stripe, and divides it into 512 B stripe segments, or *strips*. Each strip is independently checksummed and one parity strip is computed by taking the XOR of all data strips.

By design, DAX-style `mmap()` lets application modify file data without involving the file system, so it is impossible for NOVA-Fortis to keep the checksums and parity for read/write mapped pages up-to-date. Instead, NOVA-Fortis records file mapping information in inode logs and provides the following guarantee: The checksums and parity for data pages are up-to-date at all times, except when those pages are mapped read/write into an application's address space.

## 3 RESULTS

We compare NOVA-Fortis against five other file systems. Ext4-DAX, XFS-DAX and PMFS are the three DAX-enabled file systems. None of them provides strong consistency or data protection guarantees. Ext4 with data journal and Btrfs offer stronger consistency and protection guarantees, but only run on NVMM-based block devices.

We find that the extra storage NOVA-Fortis needs to provide fault-tolerance consumes 14.8% of file system space and reduces application-level performance by between 2% and 38% compared to NOVA. NOVA-Fortis outperforms DAX-enabled file systems without reliability features by 1.5× on average. It outperforms reliable, block-based file systems running on NVMM by 3× on average.

## REFERENCES

[1] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. 2009. Better I/O through byte-addressable, persistent memory. In *Proceedings of the ACM 22nd Symposium on Operating Systems Principles (SOSP '09)*.

[2] Subramanya R. Dulloor, Sanjay Kumar, Anil Keshavamurthy, Philip Lantz, Dheeraj Reddy, Rajesh Sankaran, and Jeff Jackson. 2014. System Software for Persistent Memory. In *Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14)*.

[3] Jian Xu and Steven Swanson. 2016. NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories. In *14th USENIX Conference on File and Storage Technologies (FAST '16)*.

[4] Jian Xu, Lu Zhang, Amirsaman Memaripour, Akshatha Gangadharaiah, Amit Borase, Tamires Brito Da Silva, Steven Swanson, and Andy Rudoff. 2017. NOVA-Fortis: A Fault-Tolerant Non-Volatile Main Memory File System. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*.