HasFS: A File System for NVM-based Hybrid Storage Architecture

Yubo Liu, Yutong Lu, Zhiguang Chen, Yunfei Du, Nong Xiao
Sun Yat-Sen University
National Supercomputer Center in Guangzhou

Introduction

In traditional DRAM-based file systems (such as EXT4), file systems need to periodically synchronize the dirty data on the DRAM to the disk to protect against data crash because DRAM is a volatile media. Furthermore, many file systems use journaling to guarantee the file system consistency. In our observations, periodic disk synchronization and after-crash consistency guarantee (journaling) will significantly degrade file system performance. So, we leverage NVM to tackle these two performance bottlenecks.

We present HasFS, a file system designed for the DRAM-NVM-DISK architecture. HasFS extends the main memory with NVM and treats NVM as a persistent page cache to eliminate the periodic disk synchronization overhead of dirty data. In addition, we designed an NVM-friendly, non-ordering consistency mechanism based on the hybrid storage architecture. This mechanism can provide strong consistency guarantee with low overhead. Our evaluation demonstrates that HasFS outperforms DRAM-based file systems at many workloads, even through NVM is slower than DRAM.

Background and Motivations

■ Periodic Synchronization Overhead

Journal Mode	Periodic Synchronization Cost	
	SSD	HDD
Journal Data	0.25%	0.13%
Ordered Data	75.7%	77.3%
Writeback	76.6%	78.9%

Table 1: Periodic synchronization cost.

Periodic synchronization will be triggered under some conditions, such as when the dirty page ratio exceeds the threshold or the time reaches the synchronization period. And sometimes its cost is very large (Table 1).

■ Journaling Overhead

Journal Mode	Journaling Cost	
	SSD	HDD
Journal Data	61.5%	96.8%
Ordered Data	10.3%	22.3%
Writeback	4.2%	20.7%

Table 2: Journaling cost.

Disk synchronization may break the file system consistency during a crash. Journaling is a popular consistency technique used in file systems. Our experiments (Table 2) show that the overhead of Journaling accounts for a large percentage of the file system in strong consistency mode (journal data mode).

■ Motivations

We consider NVM as a persistent page cache to avoid the periodic synchronization. We also store the journal on NVM to reduce the consistency guarantee overhead.

HasFS Design

■ Architecture

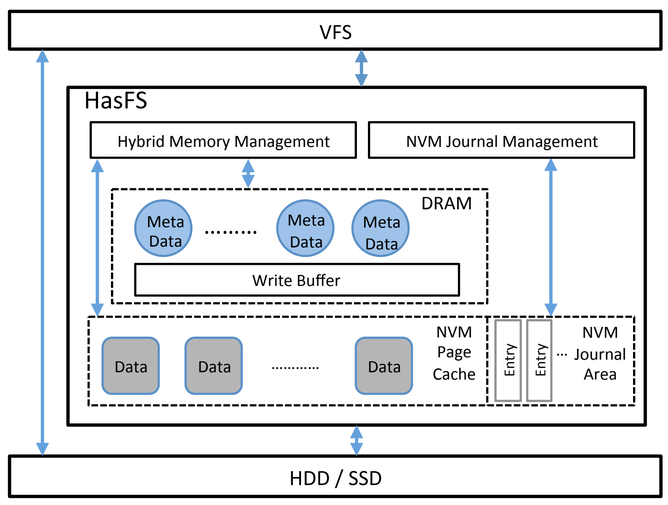


Figure 1: Architecture of HasFS

■ Non-ordering Consistency Mechanism

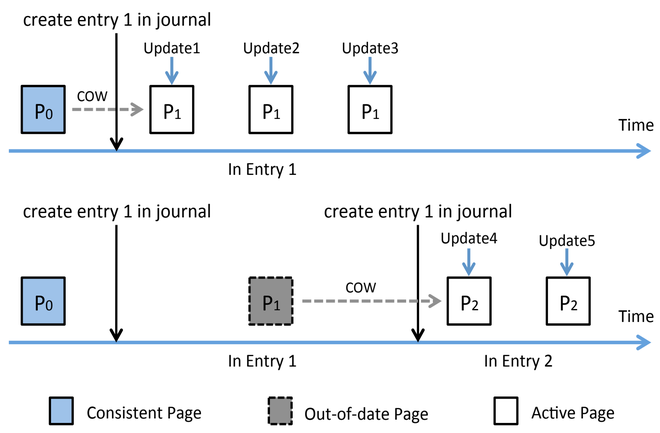


Figure 2: Lazy Copy-on-write

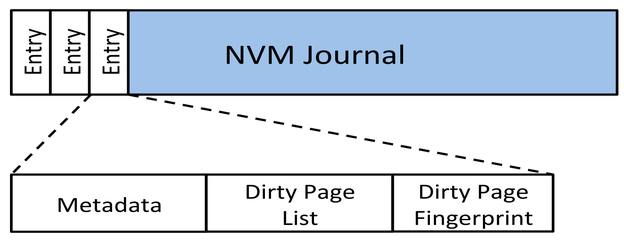


Figure 3: NVM Journal Structure

To guarantee after-crash consistency, HasFS uses lazy copy-on-write (LCOW) on data and uses non-ordering journaling on metadata.

LCOW: LCOW does not copy the page in each update. Page copy only happens when a new journal entry is generated by the file system (like Figure 5).

Non-ordering Journaling: The traditional journaling is inefficient in HasFS, because it needs to use a lot of *clflush* and *mfence* instructions to guarantee the write ordering. HasFS adds a dirty page list and a dirty page fingerprint in the journal entry (Figure 6). The dirty page fingerprint is a hash value that is calculated by the pages in the dirty pages list. After a system crash, the recovery process can calculate the fingerprint by the locations on the dirty page list to judge whether an entry is consistent.

■ Main Processes in Consistency Mechanism

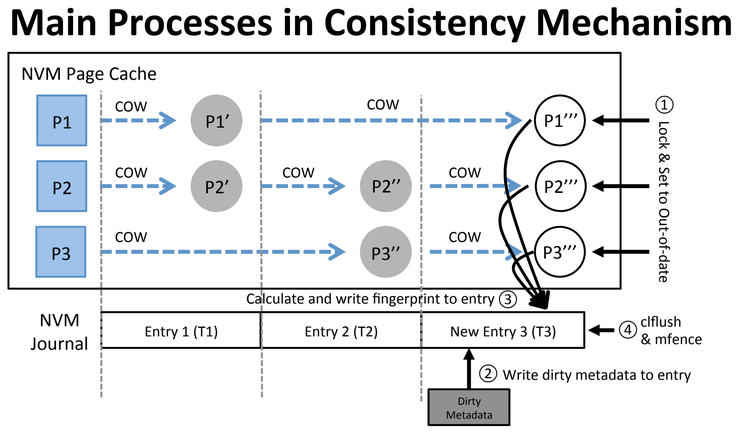


Figure 4 (a): Entry Creation

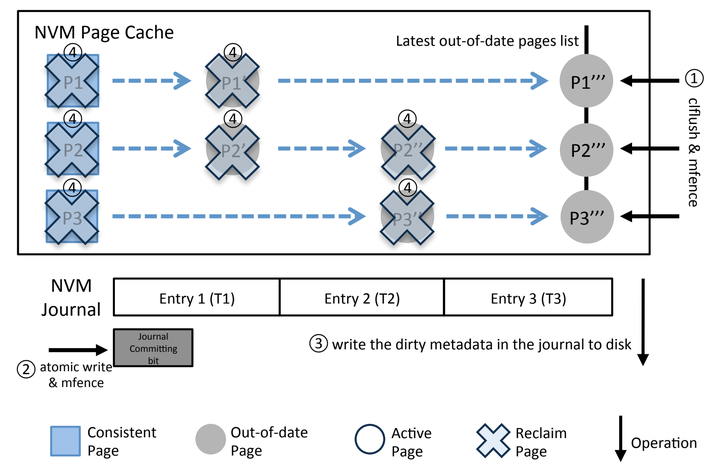


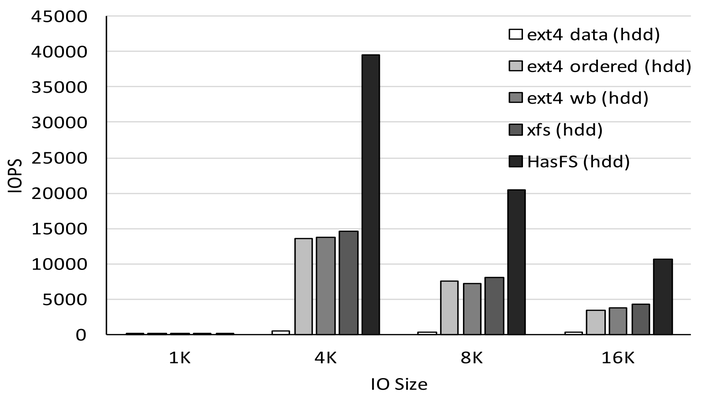
Figure 4 (b): Checkpoint

Evaluation

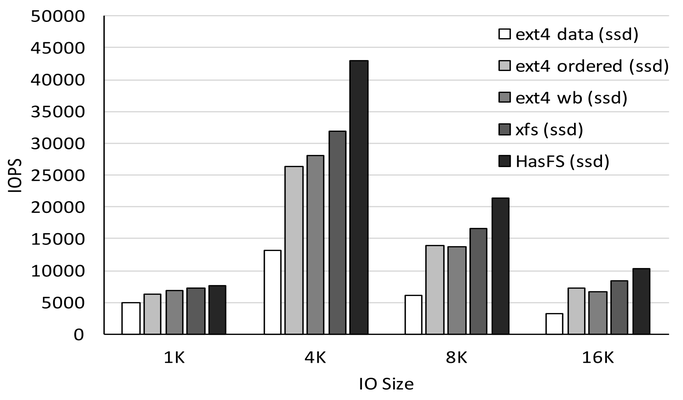
■ Experimental Setup

We ran some microbenchmarks on HasFS, XFS, original EXT4. hybrid architecture (4GB DRAM and 12GB NVM), and ran other file systems in full-DRAM architecture (16GB DRAM). And we used software to emulate NVM's write latency (200ns).

■ Early Results

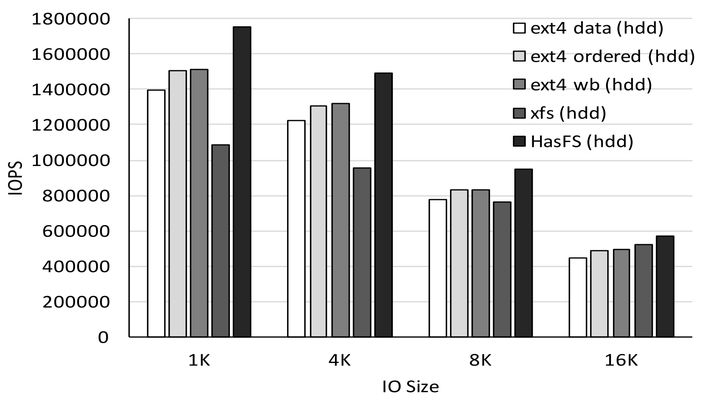


(a) Random write on HDD

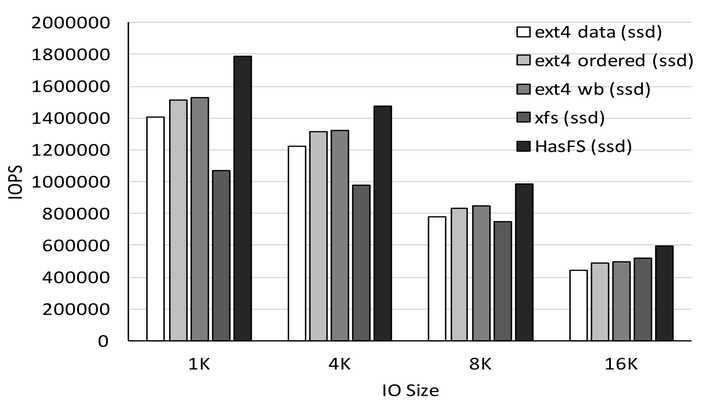


(b) Random write on SSD

Figure 5: Large File Case (20 GB)



(a) Random write on HDD



(b) Random write on SSD

Figure 6: Small File Case (1 MB)