2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress

File system for Non-volatile Main Memories: Performance Testing and Analysis

Yang Li Fang Liu Nong Xiao Songping Yu Shuo Li Yuxuan Xing
State Key Laboratory of High Performance Computing
School of Computer, National University of Defense Technology
Changsha, China
yangli.cs@outlook.com, liufang@nudt.edu.cn, nongxiao@nudt.edu.cn

***Abstract*—Emerging new memory technologies are driving the development of computer storage architectures in recent years. Non-volatile main memories (NVMMs) offer higher density than SRAM, access performance and persistency like DRAM and disk respectively. It can also persist data at memory level. However, traditional file systems are block-based and designed for the disk access features, which fail to leverage the byte-addressable characteristic of NVMMs efficiently. Consequently, there have been some NVMM file systems focusing on the study of data consistency in memory level. For instance, compared with the block-based file systems, the design of NOVA optimizes the system architecture by avoiding the page cache in DRAM and the block layer, while providing strong data consistency guarantees at the same time. In this paper, we compare state-of-the-art NVMM file systems such as PMFS and NOVA with several mainstream traditional file systems. Then, we evaluate these file systems with specific workloads and analyze the experimental results to draw some insightful conclusions. As an example, NOVA can provide write performance close to PMFS, and strong data consistency as NILFS2. When I/O size comes to 4MB, its random write performance achieves 2.2 times of Ext4, 16.5 times of NILFS2 and 4.4 times of F2FS respectively.**

Keywords-non-volatile memory; file system; performance

I. INTRODUCTION

Emerging non-volatile memory (NVM) technologies such as phase change[1], spin-torque transfer[2], Memristors[3] and Intel and Micron's 3D XPoint [5], have brought great changes for the design of storage system. NVM can promise high capacity, superior density, low latency and non-volatile compared with traditional memory, fill the performance gap between the memory and disk, and provide a new solution to the storage technology.

As non-volatile memory is both byte-addressable like DRAM and persistent like disk, it is possible to persist data at memory level using NVM. Persistent memory initially proves to be compatible with traditional file systems on a RAMDISK-like NVMM block device. However, data needs to be copied twice in both page cache and NVM, which may affect system performance significantly. As a consequence, current file systems access NVM byte-addressability by connecting NVM to memory bus directly so as to avoiding the page cache in DRAM and the block layer.

System architecture optimized for NVM has changed the traditional storage mode. It also brought about the system software changes in CPU cache model and memory management[4]. In traditional storage system, data pages are cached in DRAM and managed by operating system. Thus, it is transparent to applications whether the data page is persistent and what status it is. As the NVMM is exposed to CPU directly in persistent memory, write reordering in CPU cache will affect the data consistency in NVM. Volatile buffers and caches in CPU cores will hide latency of cache and memory access to improve performance, which will rearrange write sequences to NVM and leave data inconsistent after a power loss or system crash. Therefore, the NVMM file systems should have different designs and implementations for NVM and relevant data structures.

To solve the problem of write reordering in CPU cache mode, NVMM file systems should take measures to keep the data consistency. Intel x86 CPU architecture provides clflush [5] instruction to flush CPU cacheline, mfence to ensure the correct instruction execution order, and there are also other instructions proposed to force writes to NVM[6]. With approaches like these, CPU provides 64-bit atomic write operation to ensure write to be ordered and persistent, which is the premise of data consistency. Common solutions for consistency issues in traditional file systems such as journaling and copy-on-write[7] come from the database management system. Current NVMM-aware file systems develop a variety of technologies to balance the data consistency overhead and performance. BPFS[8] proposed Short-Circuit Shadow Paging, a partial copy-on-write technique for updating persistent data. PMFS[10] implements atomic in-place updates and undo journaling for the metadata updates. NOVA[11] uses an improved log structure and copy-on-write strategy to provide both metadata and data consistency.

In this paper, we evaluate the performance of six open-source file systems: Ext4[13], Ext4-DAX[14], NILFS2[15], F2FS[16], PMFS[10] and NOVA[11]. First, we evaluate the single read and write performance, and assess the overall performance with several representative workloads from filebench[17,18]. Then, we emulate different write latencies to evaluate NVM media impact on performance. After doing comprehensive comparisons between NVMM file systems and traditional file systems, the results are as follows:

*1)*Compared with traditional file systems, NVMM file systems perform similarly to traditional file systems when reading a single file. For write workload, we can find that in-place update is more suitable for NVM, which reduces the extra copy of the data and utilizes the random access characteristic. While the improved log structure implemented by NOVA can provide equivalent performance as in-place update and data consistency additionally.

2) For the workload including a series of files operations such as fileserver, PMFS and NOVA perform closely to each

978-1-5386-1956-8/17 $31.00 © 2017 IEEE
DOI 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.190

other, and they outperform Ext4, NILFS2, F2FS by 1.6 times, 5.2 times and 2.7 times, respectively. When Ext4 use DAX support, it performs close to PMFS with a difference of only 15%.

3) We test metadata operations in microbenchmarks and the results shows that the use of hash tree or radix tree structures generally provides better performance. For example, directory lookup in PMFS is poor, with only 87% of Ext4-DAX. NOVA keeps the radix tree in DRAM to accelerate the directory, which provides performance advantage, up to 3.9 times better than NILFS2.

4) For the read-intensive workloads, such as webserver, the experimental result shows that PMFS and NOVA perform almost the same, about 1.2x to 2x better than other file systems.

The rest of the paper is organized as follows. In Section II, the structures and characteristics of different file systems are analyzed and summarized. Section III describes our experimental methodology and different workloads. We present the results and analysis in Section IV. Finally, we provide our conclusions in Section V.

II. Background

The emergence of non-volatile memory technologies provides new solutions for computer storage technology. Byte-addressable non-volatile memories such as spin-torque transfer RAM(STT-RAM)[20], phase change memory(PCM)[21], resistive RAM(ReRAM)[22], and 3D XPoint memory technology[23] have superior density and higher integration compared with traditional memory. For example, STT-RAM, PCM and ReRAM have higher density and better scalability than DRAM. They also provide similar access latency with DRAM. Intel and Micron is developing new 3D XPoint technology which will make non-volatile memory up to 1,000 times faster than NAND. This technology will come to market in a few years.

As it can help to reduce access latency from milliseconds to microseconds or even nanosecond, non-volatile memory technology leads to the change in storage hierarchy. Using NVM instead of disk can also enhance the performance of the storage system greatly. Furthermore, it results in several new applications such as using NVM instead of DRAM, building hybrid volatile or non-volatile main memories or using DRAM caching data for better access to NVM. However, traditional file systems fail to be completely applicable to NVM mainly for the following reasons: First, traditional file systems manage data in blocks and fail to take advantage of byte-addressable feature. More importantly, most optimizations in traditional systems are designed for the sequential read and write, while NVM has little difference between random read-write operations and sequential ones. Using NVM in traditional system will also cause additional data management problems such as write amplification, space redundancy and other issues.

On account of these limitations, there appeared NVMM file systems to make full use of the byte-addressable feature of NVM and avoid overheads of block-oriented storage.

Jeremy et al. [8] proposed BPFS which placed NVM side-by-side with DRAM on the memory bus, and accessed data using ordinary CPU loads and stores. In BPFS, all persistent data structures are organized into a tree and stored in NVM. At the same time, it stores undo data blocks and hash tables in DRAM as a cache of complex data structures to accelerate the data processing and gain performance improvement. X.Wu et al. proposed SCMFS[9], which utilized the existing memory management module in the operating system to provide the allocation and management of persistent data. It allocates continuous space for each file so as to optimize the data organization. PMFS[10] proposed by Intel in 2014 maps PM pages directly into an application's address space to bypass the page caches in DRAM. It also implements a fully POSIX-compliant file system interface and other features, such as transparent large page support, to further optimize memory-mapped I/O. Xu et al.[11] implement a hybrid memory using NVM to access data and DRAM to accelerate data searching. Ou et al.[12] provide a write buffer policy to cover the shortage of the long write latency of NVMM. It buffers the lazy-persistent writes in DRAM temporarily, so as to improve the write performance.

III. File Systems Analysis

In order to enhance the understanding of NVMM file system design, we first introduce file systems such as Ext4, Ext4-DAX and NILFS2 based on hard disk. Then we introduce F2FS, a file system for flash storage. Finally we describe PMFS and NOVA designed for NVM. For the purpose or comparision we classify these file systems into two categories: traditional file systems and NVMM File Systems. A RAMDISK-like NVMM block device is used to be compatible with traditional file systems to gain benefits from memory-level persistence, as shown in Figure 1(a). NVMM file systems generally bypass the page cache and connect NVM to the memory bus, allowing the processor to access NVM directly and build persistent memory, as shown in Figure 1 (b).

Next, we introduce the design of each file system from aspects of inode data structure, update style and journaling mode.

Ext4: Ext4 has better scalability, capacity and metadata management capabilities compared to Ext3. It improves space allocation strategies, using hierarchical methods to represent small files, and trees to organize large files. What is more, it

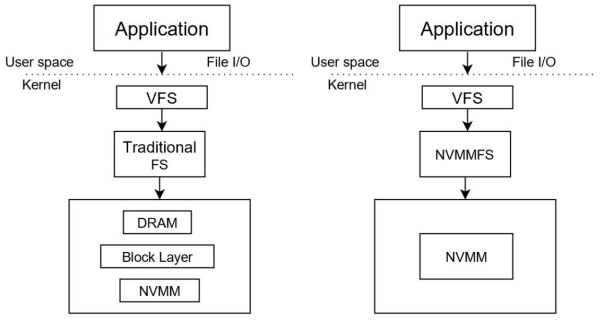

Fig.1. (a)Traditional File Systems (b)NVMM File Systems

TABLE I. COMPARISON OF FILE SYSTEMS FEATURE

	Ext4 Ext4-DAX	NILFS2	F2FS	PMFS	NOVA
Inode Data Structure	Hashed B-tree	B-tree	Multi-level hash tables	B-tree	Radix tree index
Updata Mode	In-place update	Log structure	Hybrid log structure	In-place update	Improved log structure
Data Consistency	Metadata consistency	Metadata and data consistency	Metadata consistency	Metadata consistency	Metadata and data consistency

develops further ways to improve performance, such as multiple segment assignments, pre-allocation, and delayed allocations. In terms of reliability, Ext4 supports different journaling modes, such as orderd, writeback and data. Ext4-DAX is one of its modes in which we add support to block operations to make system bypass the page cache and access the user space directly. It uses the ordered way to ensure consistency of metadata updates.

NILFS2: NILFS2 is a pure log-structured file system. Its most outstanding characteristic is the continuous snapshots which can back up the file operations. Users can simply restore the contents of the file after a system crash. However, the performance has been sacrificed to achieve system reliability. Its inodes are organized as B-tree. The metadata and data updates are appended to the end of logs to reduce the file searching and improve write performance. One major problem of this structure is that once a leaf node is updated, both direct and indirect index block have to be changed. This will cause a series of writes, which is called wandering tree.

F2FS: F2FS is a file system for flash storage, proposed by Lee et al. in 2015. It uses a NAT(Node Address Table) to serve the physical locations of all node blocks, including inodes, direct and indirect nodes. Thus, each update can only change the direct node block and its NAT entry, effectively addressing the wandering tree problem. F2FS implements two logging policies. The first one is writing the blocks to clean segments directly and transforming random write requests to sequential writes. When there is no enough free logging space, this policy starts to suffer high cleaning overheads. We turn to the second policy which requires no cleaning operations and writes blocks to obsolete space in existing dirty segments. This strategy can be well applied in flash devices.

PMFS: PMFS is proposed by Intel in 2014. It is a POSIX file system designed for exploiting PM(Persistent Memory) byte-addressable characteristic. It supports direct PM access by mapping PM pages directly into an application's address space. PMFS uses B-tree to represent the inode table and data in inodes so as to index large amounts of sparse data. But for some operations involving multiple directories, it causes a long time for directory lookup and complicated changes to the tree structure, which will cause large overhead. For consistency, PMFS uses atomic in-place updates for less than 64-byte updates, and undo journaling for metadata updates. Once the metadata is modified, the old values is first saved and appended to the log. After all the metadata updates are done, these dirty metadata will be flushed to PM and made durable. Using copy-on-write, PMFS only guarantees that the data becomes durable before the associated metadata does, the same as the guarantee provided by Ext4 in ordered data mode. As a consequence, PMFS can only provide metadata consistency.

NOVA: Xu et al. proposed NOVA, an improved log-structured file system in 2016. It stores logs as linked lists instead of allocating continuous space, which utilizes the random access characteristic and organizes data more flexibly. NOVA has per-CPU inode tables to ensure good scalability and assigns new inodes to each inode table in a round-robin order, so that inodes are evenly distributed among inode tables. This inode structure avoids the inode allocation contention and allow for parallel scanning in failure recovery. As the layout of linked lists is not suitable for searching, NOVA keeps a radix tree in DRAM for each directory inode. It hashes the dentry name as a key, and each leaf node points to the corresponding dentry in the log. By this way, NOVA not only takes the advantage of DRAM's fast access, but also ensures the efficient organization of data structure in NVM. NOVA uses 64-bit in-place writes and lightweight journaling to provide metadata atomicity. For file data, it uses copy-on-write to reduce the log size and makes garbage collection simple and efficient, and provides atomic log append by updating the log's tail pointer. By using these methods in combination, NOVA provides stronger consistency guarantee than PMFS with a small cost of performance.

Finally, we summarize the characteristics of each file system in Table 1.

IV. METHODOLOGY AND LIMITATIONS

A. Experimental Setup

We construct a performance emulator based on NVMM to simulate its performance because NVMM devices are still unavailable yet. To emulate slow writes in NVMM, we add an extra latency to each store operation to build NVMM emulator.Our emulator uses DRAM to emulate NVMM due to their similarity in performance. We configure the latency by redefine the CPU frequency and use a x86 RDTSCP instruction to read the processor timestamp and reach a definite time. As a fast NVM, STT-RAM has similar access latency and bandwith with DRAM. We emmulate it by setting the write latency to 200ns and adding the delays after the clflush instructions, and we use the same read latency and bandwidth

TABLE II. SYSTEM CONFIGURATIONS

CPU	Intel Core i5-4590,3.30GHz
CPU cores	4
Processor cache	128KB 8-way L1 1MB 8-way L2 6MB 12-way L3
DRAM	16G
NVMM	Emulated with slowdown the write latency is 200 ns
Operating System	Ubuntu 16.04, kernel version 4.4.0

as DRAM. To compare with traditional block-based file systems, we also construct a NVMMBD emulator to emulate the NVMM-based block device by modifying Linux's ramdisk in brd device driver to build the performance model. We evaluate several file systems described in the previous section, including Ext4, Ext4-DAX, NILFS2, F2FS, PMFS and NOVA. The first four are block-based file systems running on the NVMMBD emulator, and the latter two can access the NVMM directly, so they run directly on the NVMM performance simulator. As NOVA is the latest NVMM file system, we will use it as a reference to each file system in the following analysis and evaluation. The system configurations are shown in Table 2.

B. Workloads

In order to evaluate the performance of these file systems under different workload parameters,we selected not only microbenchmarks, but also four common predefined workloads from Filebench[17,18]: Fileserver, Webserver, Webproxy and Varmail. Specific characteristics are shown in Table 3.

Specifically, Fileserver emulates a server with massive directories of multiple users. Each thread representing a user will select a series of files and do a list of operations expected from a real file-server such as creates, deletes, reads, writes, appends and status. To emulate HTTP requests, each thread in Webserver opens a file, reads it in one call, then closes the file. Every 10th read, it appends a small amount of data to a log file. File sizes follow a gamma distribution, with an average file size of 16 KB. Webproxy uses a large number of threads to mix a series of create, add, read and delete operations. It is distinguished by its flat namespace, concluding almost all files in a large directory. Varmail workload represents a workload experienced by a /var/mail directory that uses Maildir format (one message per file). When a user receives an email, a file is created, written, and fsynced. It also includes the reread oprations. Average email size is defined as 16 KB.

TABLE III. FILEBENCH WORKLOAD CHARACTERISTICS

Workload	Average file size	I/O size (r/w)	Threads	R/W ratio	Metadata operations
Fileserver	128KB	16KB/16B	50	1:2	Yes
Webproxy	16KB	1MB/16KB	100	5:1	Yes
Webserver	16KB	1MB/16KB	100	10:1	No
Varmail	16KB	1MB/16KB	16	1:1	Yes

C. Limitations

In this work, we evaluate NVMM file systems only from the perspective of performance. We do not examine other features such as instant durability, recovery point and media endurance. NVM main memory can provide comparable performance with low energy consumption to that of a DRAM main memory. However, it is necessary to take into account the endurance levels of NVM technologies when designing NVMM file systems. As an example, PCM can endure few millions to hundreds of millions of writes per cell. Even though it is far more wear-resisting than NAND-based Flash, the PCM is considered worn-out when the writes to one bit exceeds the endurance limit [19]. Therefore, the specific cells may wear out more possibly for the applications modifying the same chunk of data intensively. PMFS uses in-place updates which may accelerate the single cell wastage. While NOVA uses copy-on-write to ensure data consistency, which increases the total amount of data writing.

V. EVALUATIONS

A. Read and Write

First, we measure read performance of these file systems in different I/O size. Figures 2(a) and (b) describe the results of sequential and random reads respectively for a single large file with I/O sizes from 1KB to 4MB in four threads. We can find that traditional file systems and NVMM file systems perform similarly in read tests. When I/O size is less than 64KB, Ext4 performs better than Ext4 due to its page cache. While Ext4-DAX can not hit the cache and software overhead has a larger impact on performance. As we increase the I/O size, the advantage of Ext4-DAX gradually appears. Ext4-DAX outperforms other file systems by up to 1.4 times in random read workload when I/O size is 4M.

Figure 3 (a) and (b) show the sequential and random write throughput when we varied the I/O size from 1KB to 4MB. On behalf of traditional file systems, Ext4 provides steady performance using a in-place update mode. When I/O size is small, it benefits from page cache and performs only second to PMFS. As I/O size is gradually increased, it tends to be stable due to the double copy of data. Nevertheless, when I/O size is larger than 4KB, Ext4-DAX performs close to PMFS, which indicates that adding DAX support to traditional file system can help improve performance and utilize NVM better. NILFS2 has the worst performance in the write-intensive workload because of its continuous snapshots, and it gains strong system reliability at this price. As a log-structured file system, F2FS has better performance than NILFS2, but is inferior to others. This is because the journal will be filled up quickly due to the fast access of NVM. This will cause frequent flush operations and decrease performance. PMFS, a representative file system designed for NVM features, has changed the infrastructure of traditional file systems. It supports direct access to NVM, atomic updates and fine-grained logging, which is a great benefit to performance. As a

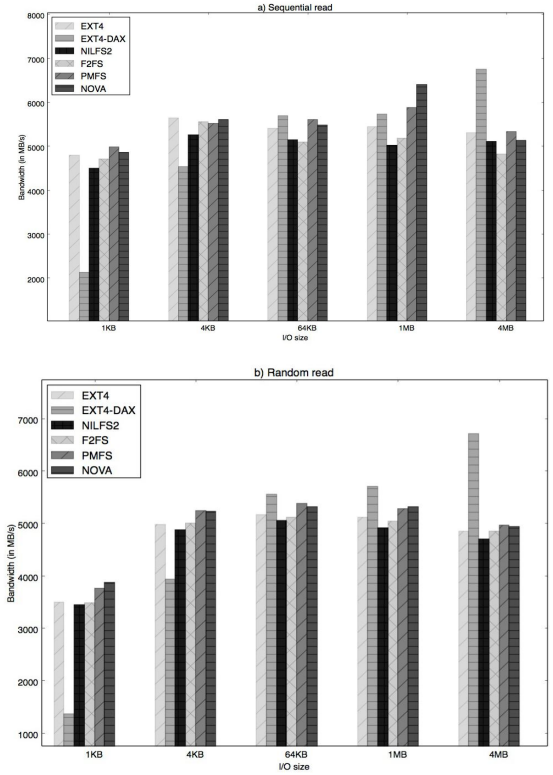

Fig.2 Read throughput with different I/O size

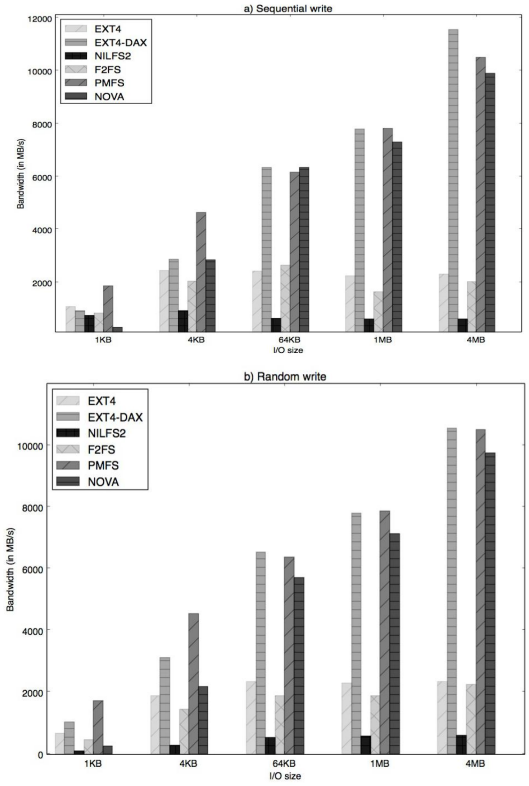

Fig.3 Write throughput with different I/O size

result, when I/O size is 1KB, it can reach 2.6 times to Ext4, 22 times to NILFS2 and 3.8 times to F2FS. When I/O size rises to 4MB, it comes to 4.5x, 17x and 4.7x compared to Ext4, NILFS2 and F2FS respectively. NOVA improves the traditional log structure, and uses copy-on-write technology to update data. When I/O granularity is less than 4KB, there exists write amplification problem, leading to its lowest performance. As I/O size grows, it provides high write performance close to PMFS and strong data consistency as NILFS at the same time. When I/O is 4MB, it turns out to be 2.2x, 16.5x and 4.4x better than Ext4, NILFS2 and F2FS respectively.

For read and write workloads, we can find that in-place update is more suitable for NVM, which reduces the extra copy of the data and utilizes the random access characteristic. The improved log structure implemented by NOVA can provide equivalent performance as in-place update and data consistency additionally. Therefore, the key is to fully develop NVM characteristics and utilize them reasonably.

B. Macrobenchmarks

Fig. 4 shows the results of fileserver workload on NVM. Fileserver includes a series of create, write, and append operations. We can see that NILFS2 performs worst because of the wandering tree problem. When writing a file or creating a directory, the recursive updates lead to low performance. F2FS solved this problem by introducing NAT (Node address Table). F2FS has three kinds of metadata including inode, direct dnode and indirect dnode. Direct dnode points to the data block address, and indirect dnode refers to the entry in NAT table,

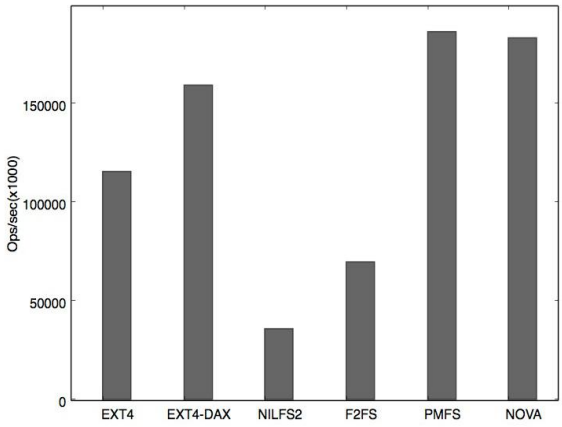

Fig.4 Performance of file systems under fileserver workload

which includes inode number and block address. Therefore, when the data changes, only the direct dnode and NAT table address need to be modified. As shown in the figure, F2FS outperforms NILFS2 by 80%. PMFS and NOVA perform close to each other, better than Ext4, NILFS2 and F2FS by a factor of 1.6, 5.2 and 2.7. We see an improvement of 38% when adding DAX support to Ext4, only 15% difference with NVMM file systems.

Webproxy involves multiple metadata operations and can evaluate the lookup performance in a large directory. As shown in Fig. 5, Ext4-DAX performs 15% better than PMFS benefiting from its hashed B-tree structure. Ext4 also performs

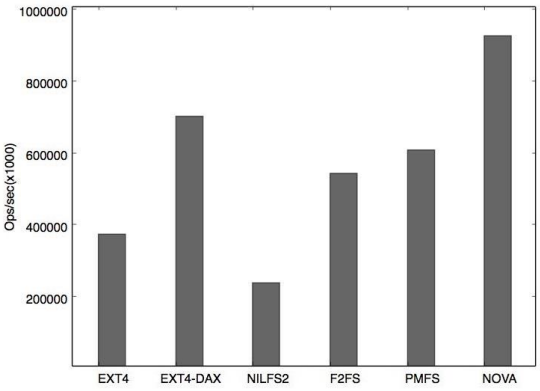

Fig.5 Performance of file systems under webproxy workload

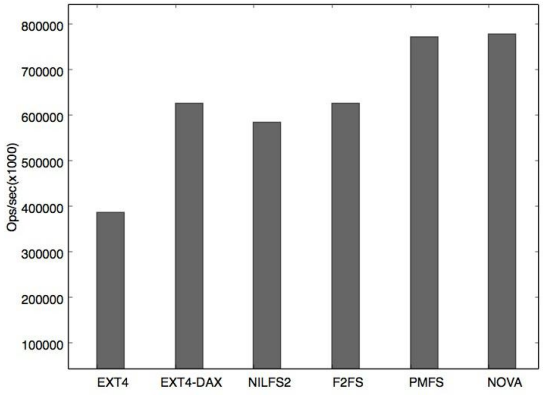

Fig.6 Performance of file systems under webserver workload

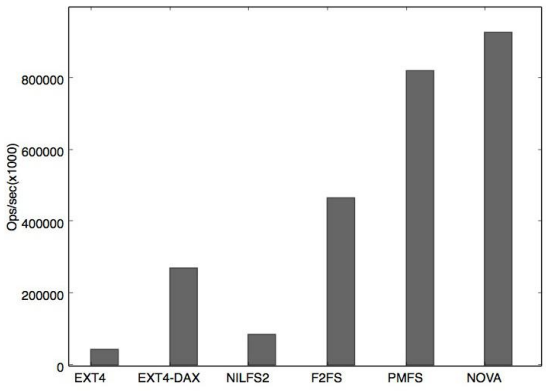

Fig.7 Performance of file systems under varmail workload

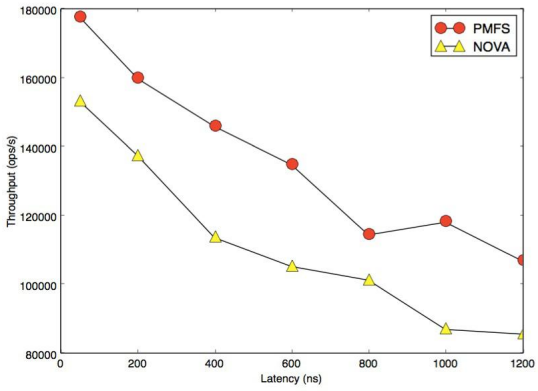

Fig.8 Throughput for Different NVMM Write Latencies.

well, about 1.6 times better than NILFS2. This proved that adding hash strategies to ordinary tree structures can improve the lookup performance apparently. F2FS uses multi-level hash tables to organize directories and it proves to perform 2.38 times better than NILFS2. PMFS benefits from its system architecture, but is inferior to other two file systems which have the same direct access feature. NOVA keeps a radix tree in DRAM for each directory inode to speed up dentry lookups. It also maintains separate logs for each inode to support massive concurrency. As a result, it performs about 3.9 times compared to NILFS2, and 1.5 times better than PMFS, the best across all the file systems.

Webserver is a read-intensive workload, involving some lookup operations. The result of each file system is shown in Figure 6. Same as the previous experiments, PMFS and NOVA achieve similar results in experiments due to the same read path. Also they perform about 100%, 30% and 20% better compared to Ext4, NILFS2 and F2FS. Ext4-DAX is just behind these two file systems, around 80% of PMFS. This result is mainly due to the elimination of the additional memory copy and it also proves the performance improvement of traditional file systems in DAX mode.

Varmail emulates an email server with a large number of small files, involving both read and write operations. As described in Figure 7, NOVA performs best among these file systems, about 13% better than PMFS. This is the same as the tests above as NOVA performs better in write workload. NILFS2 and F2FS performed 10.9x and 2x worse compared to NOVA. Ext4-DAX has the similar performance with PMFS, outperforming 6x than Ext4. These results are consistent with our previous analysis of single read and write workloads.

When using a traditional file system on NVM, we can add a DAX support to improve the performance. NVMM file systems can utilize the byte-addressable characteristic and low latency of NVMM such as PMFS. To achieve better performance, more optimizations on concurrency, atomic file operations and efficient garbage collection should be implemented as the design in NOVA, providing the strong data consistency at the same time.

The NVMM write latency can also affect the system performance. Figure 8 shows the throughput performance when we vary the NVMM write latency from 50 ns to 1200 ns using a single thread on fileserver workload. We can observe that PMFS performs better than NOVA no matter how much the latency would be. The performance gap are larger than that performs in Figure 4. This is because NOVA can leverage the concurrency as its inodes are assigned to different logs, narrowing the performanve gap when the thread number rises to 50. We can see that as the write latency is increased to 1200ns, the throughput of PMFS and NOVA has reduced by 40% and 45%. As all the data are written to NVM directly, the performance impact of write latency becomes more obviously. To minimize the impact of NVM long write latency, there is a

strategy of using DRAM as a write buffer proposed by Ou in HiNFS[12].

For these macrobenchmarks, we make the following conclusions through the measurements and observations:

1) We find that traditional file systems such as Ext4 can use DAX support to pass the page caching and reduce software overhead. We have shown the benefits that Ext4-DAX performs close to NVMM file systems with only 15% difference in fileserver workload, 30% in webproxy workload, and 20% in webserver workload.

2) PMFS owns a stable and excellent performance in most cases except directory lookup in webproxy, compared to other five file systems.

3) The latest NVMM file system, NOVA, can provide better performance in read and write operations compared to traditional file systems. Although it might be slightly unsatisfactory in small I/O sizes because of write amplification problem. Compared to the current existing file system, when I/O reaches 4MB, it can perform 2.2x, 16.5x and 4.4x better than Ext4, NILFS2 and F2FS respectively.

VI. Related Work

Lee et al. [24] explored the use of NVM storage from the operating system's perspective. They emulated NVM by DRAM and compared it with that of HDD storage environments under various configurations, such as buffer caching, synchronous I/O, direct I/O, etc, using Ext4 file system for all the evaluations. Sehgal et al.[25] evaluated the performance of various legacy Linux file systems under various real world workloads and compared it against PMFS. They concentrated on evaluating different file system configurations in mount and format options. In this paper, we perform our experiments on ramdisk and simulate NVM write latency to test the impact of NVM media more accurately. We choose four typical traditional file systems and two advanced NVMM file systems to evaluate how these different designs perform on NVM. We use NOVA, the latest NVMM file system, as a reference to each file system and compare it with PMFS.

VII. Conclusion

In this paper, we evaluate the performance of six mainstream file systems including traditional file systems and advanced NVMM file systems on the NVMM emulator. We run different file systems under varying workloads and do comprehensive comparison between these file systems. After we discuss the implications of the measurement study performed in this paper, we can find that some of the strategies of the traditional file systems are unable to utilize the byte-addressable feature of NVM. Although we test it at memory level, the performance gap between these two kinds of file systems still exists. As the latest NVMM file system, NOVA can provide write performance close to PMFS, and strong data consistency as NILFS2. When I/O size comes to 4MB, its random write performance achieves 2.2 times of Ext4, 16.5 times of NILFS2 and 4.4 times of F2FS respectively.

In general, the strategies of existing file systems possess corresponding advantages and disadvantages in difference application scenarios. We hope to make more reasonable utilization of NVM, and further optimize NVMM file systems based on the analysis and conclusions in the future.

Acknowledgment

This work is supported by National Natural Science Foundation of China (61433019, U1435217) and The National Key Research and Development Program of China (2016YFB1000302).

References

[1] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative" in ACM SIGARCH Computer Architecture News, vol. 37, no. 3. ACM, 2009, pp. 2 – 13.

[2] Y. Huai. Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects. AAPPS Bulletin, 18(6):33 – 40, Dec. 2008.

[3] F. Xu-Dong, T. Yu-Hua, and W. Jun-Jie, "Spice modeling of memristors with multilevel resistance states," Chinese Physics B, vol. 21, no.9, p.098901, 2012. https://www.micron.com/about/emerging - technologies/3d-xpoint-technolgy

[4] S. Yu, N. Xiao, M. Deng, F. Liu, and W. Chen, "Redesign the memory allocator for non-volatile main memory"， ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 13, no. 3, p. 49, 2017.

[5] I. Intel, "Intel-64 and ia-32 architectures software developer's manual," Volume 3A: System Programming Guide, Part, vol. 1, no. 64, 2013.

[6] I. Cooperation, "Intel architecture instruction set extensions programming reference," 2016.

[7] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, Operating systems: Three easy pieces. Arpaci-Dusseau Books Wisconsin, 2014, vol. 151.

[8] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, "Better i/o through byte-addressable, persistent memory," in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, 2009, pp. 133 – 146.

[9] X. Wu and A. L. N. Reddy. SCMFS: A File System for Storage Class Memory. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC ' 11, pages 39:1 – 39:11, New York, NY, USA, 2011. ACM.

[10] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jackson, "System software for persistent memory,"in Proceedings of the Ninth European Conference on Computer Systems. ACM, 2014, p. 15.

[11] J. Xu and S. Swanson, "Nova: A log-structured file system for hybrid volatile/non-volatile main memories." in FAST, 2016, pp. 323 – 338.

[12] J. Ou, J. Shu, and Y. Lu, "A high performance file system for non-volatile main memory," in Proceedings of the Eleventh European Conference on Computer Systems. ACM, 2016, p. 12.

[13] M. Cao, S. Bhattacharya, and T. Ts' o, "Ext4: The next generation of ext2/3 filesystem." in LSF, 2007.

[14] M. Wilcox, "Add support for nv-dimms to ext4."

[15] B. A. Dhas, E. Zadok, J. Borden, and J. Malina, "Evaluation of nilfs2 for shingled magnetic recording (smr) disks," Stony Brook Univ. And Western Digital Corp., Stony Brook, NY, Tech. Rep. FSL-14-03, 2014.

[16] C. Lee, D. Sim, J. Y. H16wang, and S. Cho, "F2fs: A new file system for flash storage." in FAST, 2015, pp. 273 – 286.

[17] Filebench file system benchmark. Available: http://sourceforge. net/projects/filebench

[18] Filebench: A Flexible Framework for File System Benchmarking

[19] Awad A, Blagodurov S, Yan S. Write-Aware Management of NVM-based Memory Extensions[C]// International Conference on Supercomputing. ACM, 2016:1-12.

[20] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-effcient main memory alternative," in

Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on. IEEE, 2013, pp. 256 – 267.

[21] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in ACM SIGARCH Computer Architecture News, vol. 37, no. 3. ACM, 2009, pp. 2 – 13.

[22] R. Fackenthal et al., "A 16gb reram with 200mb/s write and 1gb/s read in 27nm technology, isscc tech," 2014

[23] I. Newsroom, "Intel and micron produce breakthrough memory technology, july 28, 2015."

[24] E. Lee, H. Bahn, S. Yoo, and S.H Noh. Empirical Study of NVM Storage: An Operating System's Perspective and Implications. In IEEE 22nd International Symposium of Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2014.

[25] Sehgal P, Basu S, Srinivasan K, et al. An empirical study of file systems on NVM[C]// MASS Storage Systems and Technologies. IEEE, 2015:1-14.