JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.17, NO.1, FEBRUARY, 2017 ISSN(Print) 1598-1657
https://doi.org/10.5573/JSTS.2017.17.1.001 ISSN(Online) 2233-4866

Optimized Adoption of NVM Storage by Considering Workload Characteristics

Jisun Kim and Hyokyung Bahn

Abstract—**This paper presents an optimized adoption of NVM for the storage system of heterogeneous applications. Our analysis shows that a bulk of I/O does not happen on a single storage partition, but it is varied significantly for different application categories. In particular, journaling I/O accounts for a dominant portion of total I/O in DB applications like OLTP, whereas swap I/O accounts for a large portion of I/O in graph visualization applications, and file I/O accounts for a large portion in web browsers and multimedia players. Based on these observations, we argue that maximizing the performance gain with NVM is not obtained by fixing it as a specific storage partition but varied widely for different applications. Specifically, for graph visualization, DB, and multimedia player applications, using NVM as a swap, a journal, and a file system partitions, respectively, performs well. Our optimized adoption of NVM improves the storage performance by 10-61%.**

Index Terms—**Non-volatile memory, storage system, swap device, file system, journaling**

I. INTRODUCTION

High performance NVM (non-volatile memory) media such as PCM (phase-change memory), STT-RAM (spin transfer torque RAM), and 3D Xpoint are anticipated to be adopted in the design of future computer systems. Specifically, NVM is expected to be used as secondary storage in addition to flash memory or HDDs (hard disk drives) due to their desirable properties such as relatively high performance, low-power consumption, and long endurance cycle [1-6]. This paper investigates how much performance gain can be obtained if we add NVM as various storage components (e.g., journal device, swap device, or file system device) of computer systems.

NVM has also been considered as a strong candidate to replace DRAM as it is a byte-addressable medium and has substantial density benefits over DRAM. However, it does not show competitive performance to be a memory medium in current hardware specifications. In particular, the access time of PCM is slower than DRAM about 2-5x in reads and 8-50x in writes [7, 8]. Hence, it is recently being considered as a high-speed storage rather than a memory medium.

NVM hardware technology has already reached a certain level of maturity. Specifically, PCM has been commercialized and equipped in certain types of smartphones. Patents published recently by Intel describe a detailed micro-architecture to support PCM as memory and/or a storage device, implying that NVM based computer architectures are imminent [9, 10]. The primary interfaces for NVM is likely to be DIMM or PCI-e rather than other block I/O interfaces. This is because existing block I/O interfaces such as SATA or SAS are not fast enough to support high-performance NVM devices, limiting the full advantages that NVM conveys.

Our aim is to maximize the benefit of NVM if we add NVM to the storage component of computer systems. To do this, we first analyze storage I/O requests under various applications. There are three types of storage I/O

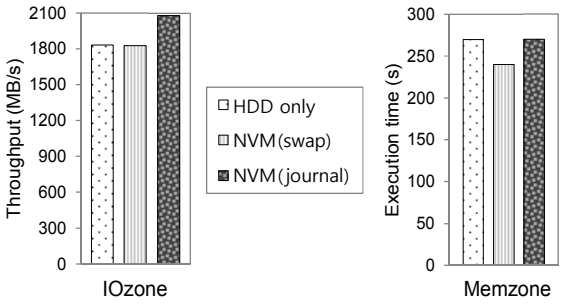
Manuscript received Oct. 18, 2016; accepted Jan. 15, 2017
A part of this work was presented in Korean Conference of Semiconductors, Seoul in Korea, Feb. 2016.
Department of Computer Science & Engineering, EWHA Womans University, Seoul 120-750, Korea
E-mail : bahn@ewha.ac.kr

requests that occur in computer systems: file system I/O, journaling I/O, and swap I/O. As these three types of I/O happen on different storage partitions, we can adopt NVM as a file system partition, a journaling partition, or a swap partition. As we have only limited NVM capacity, we should determine which partition of the storage area will be composed of NVM.

Our analysis shows that a bulk of I/O does not happen on a single specific partition, but it is varied significantly for different application categories. In particular, journaling I/O accounts for a dominant portion of total I/O in DB applications like OLTP as transaction managements in these applications require consistent writes to the journal area. In contrast, swap I/O accounts for a large portion of I/O in graph visualization applications like gnuplot as such applications exhibit large memory footprint. That is, as the memory capacity is not sufficient to accommodate the working-set of the application, swapping occurs. For other applications such as web browsers and multimedia players, file system I/O accounts for a large portion of total I/O as these applications require consistent data file accesses.

Based on these observations, we estimate how much performance gain can be obtained if we adopt NVM as different storage components and find out that the best performance through adding NVM is not obtained by fixing it as a specific storage partition but varied widely for different applications. Specifically, for graph visualization applications, using NVM as a swap device performs the best. In contrast, for DB applications, using NVM as a journal device performs well, in general, but assigning a small portion of NVM to a file system partition in order to absorb hot file I/O performs even better. For other applications such as web browser and multimedia players, using NVM as a file system partition performs the best. The performance improvement of the proposed solution is in the range of 10-61% in comparison with the system architecture that does not use NVM storage.

The remainder of this paper is organized as follows. Section II describes the motivation of this research. Section III analyzes storage I/O requests under various application categories. An optimized adoption of NVM based on the analysis and the performance evaluation results are given in Section IV. Finally, we conclude this paper in Section V.



(a) IOzone Results (b) Memzone Results

Fig. 1. Benchmark Results.

II. MOTIVATIONS

In this section, we perform real measurement studies to investigate how much performance gain can be obtained if we add NVM as a swap or a journal device. Our experiments are performed on Linux kernel 3.16.0 and Ext4. As commercially available NVM hardware is limited, we emulate it by making use of DRAM on DIMM slots with appropriate timing delays. As we want to use NVM as a swap or a journal device, it should be recognized as a block I/O device. Thus, we develop an NVM device driver based on the existing Ramdisk driver. We measure the performance of the original system that uses HDD only and new systems that additionally use NVM as swap and journal devices, which we call NVM(swap) and NVM(journal), respectively. We run two benchmarks: IOzone and Memzone for I/O and memory intensive workloads, respectively.

Fig. 1(a) shows the measured throughput of IOzone with the three architectures. As shown in the figure, NVM(journal) performs the best as it performs journaling I/O on NVM instead of slow storage. NVM(swap) does not exhibit such good results as it has the effect of extending memory capacity but IOzone is an I/O-intensive workload. Fig. 1(b) shows the measured execution time when Memzone is run. The results show that NVM(swap) significantly improves the performance of the system. Though the memory size itself is not extended, NVM(swap) performs well in memory-intensive workloads as it provides a high performance swap device.

This preliminary results show that the maximum performance gain of NVM cannot be obtained if we use it as a single fixed partition. In the next section, we will

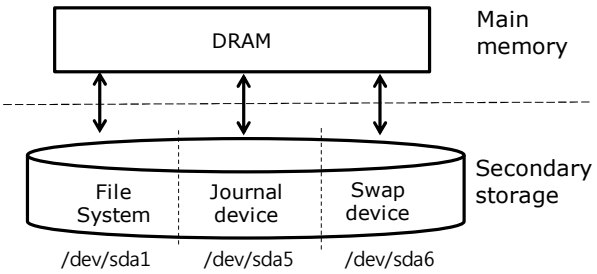


Fig. 2. Partition setting for trace collection.

Table 1. Storage partition information

Partition	Start	End	Description
/dev/sda1	2,048	103,999,487	Filesystem
/dev/sda5	104,001,536	111,998,975	Journal device
/dev/sda6	112,001,024	119,998,463	Swap device

analyze storage I/O requests on various application categories with respect to file system I/O, journaling I/O, and swap I/O, and then estimate how much performance gain can be obtained for each application category and I/O types if we use NVM.

III. Analysis of Application I/O Traces

We collect storage I/O traces while executing various application categories and analyze them. To see the relative intensity of different storage area, we partition a HDD device into a file system, a journal area, and a swap area as shown in Fig. 2. Our experimental setting consists of 512GB DDR3-10700 memory and 60GB SATA HDD. For OS and file system, we use Ubuntu 14.04 64 bit and Ext4, respectively. Note that we use the external journaling option of Ext4 to collect the journal I/O separately. Table 1 shows the details of each partition we set.

We use four application categories: gnuplot a graph drawing tool, videoplayer a multimedia player application, sysbench an OLTP workload generating benchmark, and firefox a web browser. Fig. 3 shows the amount of I/O generated on each storage partition for the four applications. As shown in the figure, a bulk of I/O does not happen on a single specific partition, but it is varied significantly for different application categories.

Specifically, swap I/O accounts for a large portion of I/O in gnuplot as shown in Fig. 3(a). Graph visualization applications require large memory to calculate and draw

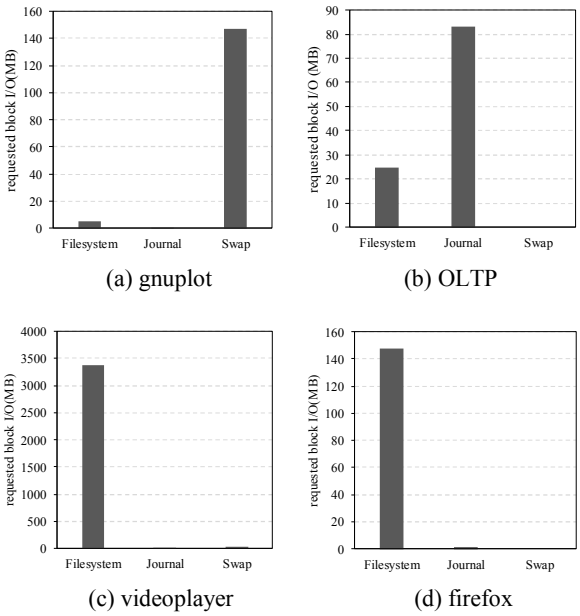


Fig. 3. I/O requests that occur on each storage partition for different applications.

each point in a graph, and thus the memory capacity may not be sufficient to accommodate the working-set of the application, which subsequently incurs large swap I/Os.

In contrast, journaling I/O accounts for a dominant portion of total I/O in OLTP as shown in Fig. 3(b). As OLTP has a sequence of transaction processing workloads, it generates large I/Os on the journal area. However, OLTP also generates a substantial amount of I/O on the file system partition. Although journaling is used in transaction processing applications, file system I/O also accounts for a certain portion as files and their metadata should also be accessed to perform these kind of applications.

Unlike the aforementioned applications, Fig. 3(c) and (d) exhibit large I/O requests on file system partitions. As a video player consistently reads a data file to play from the file system, it incurs large read I/Os from the file system as shown in Fig. 3(c). Similarly, firefox generates a dominant portion of file system I/O as shown in Fig. 3(d). Web browsers usually read a web object from the web server and cache it on the local file system, which incurs file system I/O. The stored file will be used again if the user requests the same web page in the near future.

Fig. 4 shows the distinct I/O requests that occur on each storage partition for the same application traces of Fig. 3. That is, we count only once for the same sector address although it was accessed multiple times in Fig. 4.

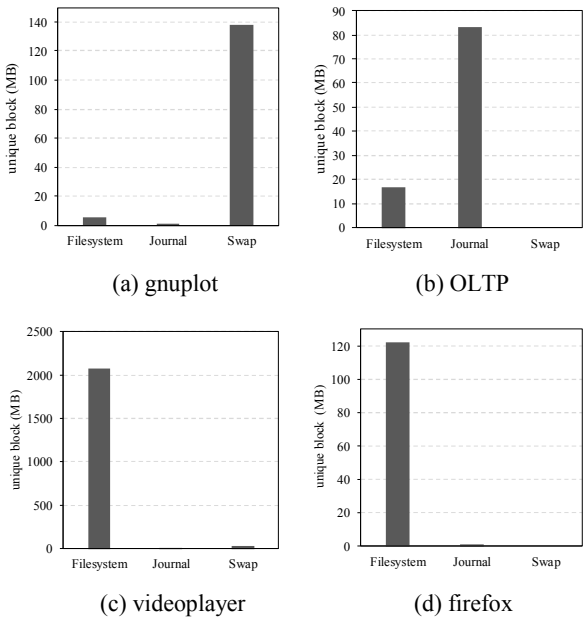


Fig. 4. Distinct I/O requests that occur on each storage partition for different applications.

By comparing the corresponding graphs in Fig. 3 and 4, we can estimate the relative hotness of each partition. For example, if the height of a graph in Fig. 4 is relatively lower than the height of the corresponding graph in Fig. 3, that partition is certain to have hot I/O sectors. In other words, some sectors in that parition have essentially been accessed multiple times.

Now, let us see in Fig. 3 and 4 which partitions have such hot I/O sectors. As the shapes of Fig. 4(a), (c) and (d) are similar to those of Fig. 3(a), (c) and (d), respectively, we cannot observe relatively hot I/O sectors from gnuplot, videoplay, and firefox. However, the height of file system partition in Fig. 4(b) is relatively lower than that of Fig. 3(b). This implies that OLTP has some hot I/O sectors on the file system partition although journal area has more I/O requests.

IV. OPTIMIZED ADOPTION OF NVM STORAGE

In this section, we discuss how NVM storage can be used appropriately for reflecting the I/O characteristics analyzed in Section III. Fig. 5 shows the traditional system architecture and the target architecture of this research. NVM is adopted as a part of storage layer, which can be used as a file system, a journal area, or a swap area, according to the application characteristics we analyzed.

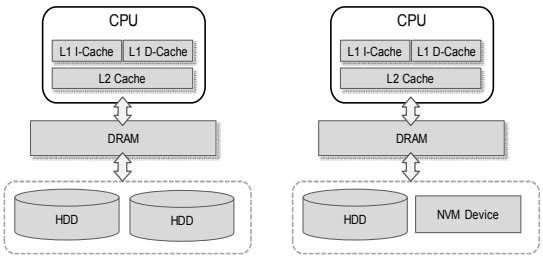


Fig. 5. Storage architecture with NVM.

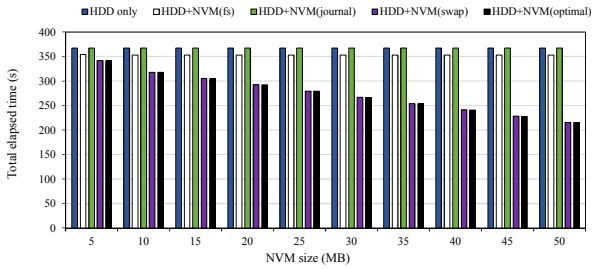
As we have only limited NVM capacity, we should determine which partition of the storage area will be composed of NVM. We analyzed total I/O requests and total distinct requests in Section III, with which we can estimate how much performance gain can be obtained if we adopt NVM as different storage components.

NVM storage that can be adopted in our study is not limited to some specific types, but any of PCM, STT-MRAM, or FeRAM can be utilized. However, we think that PCM is the most realistic medium as it is prospected to be fast storage but substantially slower than DRAM [11, 12]. STT-MRAM is suitable for main memory rather than storage as its access latency is as fast as that of DRAM and is also byte-addressable. FeRAM is difficult to enhance its density any longer. Due to this reason, we set PCM as the default type of NVM in our experiments.

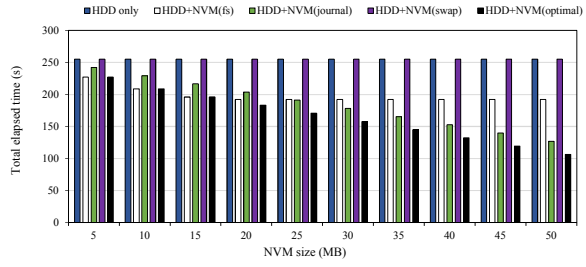
Fig. 6 shows the total elapsed time for executing gnuplot, OLTP, videoplayer, and firefox when we use the original HDD only and a small amount of additional NVM. The figure also presents the performance of an offline optimal placement with the given NVM capacity. To do so, we sort blocks by their access counts after executing the applications, and then place top blocks with the largest access count on NVM first regardless of their original partitions (i.e., file system, journal, or swap), while the remaining blocks with relatively small access counts are placed on HDD. We call this HDD+NVM(optimal), which is not a practical solution but displayed for comparison purposes.

As shown in the figure, the best performance through adding NVM is not obtained by fixing it as a specific storage partition but varied widely for different applications. Another interesting observation is that we can decide the partition of NVM appropriately, thereby obtaining performances similar to that of offline optimal placement.

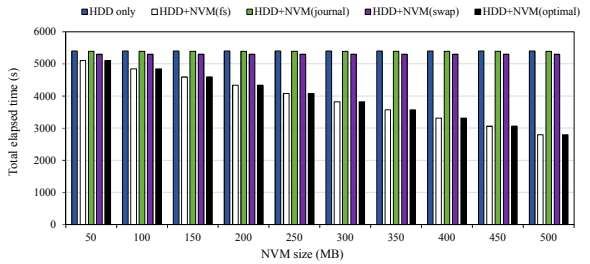
As shown in Fig. 6(a), in case of gnuplot, using NVM



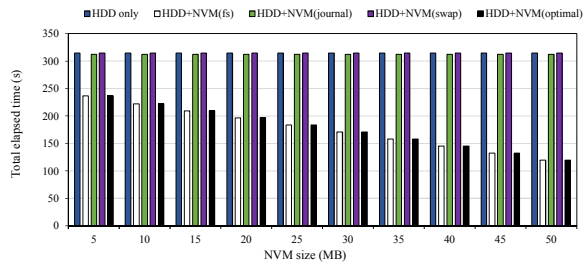
(a) gnuplot



(b) OLTP



(c) videoplayer



(d) firefox

Fig. 6. Total elapsed time of HDD, HDD+NVM(fs), HDD+NVM(journal), HDD+NVM(swap) and HDD+NVM (optimal) as NVM capacity is varied.

as a swap device consistently performs better than other placements and it almost approaches the offline optimal placement. This is consistent with Fig. 3(a) and 4(a) that exhibit a large swap I/O.

Fig. 6(b) shows the OLTP results, which contrast significantly as the NVM capacity is varied. When the NVM capacity is less than 25MB, HDD+NVM(fs) performs the best. However, as the NVM capacity increases, the performance of HDD+NVM(journal) is improved sharply, whereas HDD+NVM(fs) does not show improvement any longer. This is also consistent with the analysis results in Section III. That is, OLTP has some hot I/O sectors on the file system partition, which can be absorbed by assigning a small NVM partition. Also, as OLTP has a bulk of I/O on journal partitions, we can absorb them by assigning subsequent NVM capacity to the journal area.

Fig. 6(c) and (d) show the videoplayer and firefox results. The performances of these two applications can be improved significantly when we adopt NVM as a file system partition. In particular, the total elapsed time is improved by 7-50% as the NVM capacity increases. This is also consistent with the analysis results in Section III. Also, adopting NVM as file system partitions exhibits similar results to that of offline optimal placement in these applications.

We can summarize the suggestions of using NVM from the results as follows. For graph visualization, DB, and multimedia player applications, we suggest NVM to use as a swap, a journal, and a file system partitions, respectively. For DB applications such as OLTP, however, performances can be improved even better if we use a small portion of NVM as a hot file partition.

V. CONCLUSIONS

In this paper, we presented how NVM storage can be adopted efficiently for different application categories. We anatomized storage I/O requests with respect to file system I/O, journaling I/O, and swap I/O under various applications and observed that a bulk of I/O does not happen on a single storage partition, but it is varied significantly for different application categories. In particular, journaling I/O and swap I/O account for a dominant portion of total I/O in DB and graph applications, respectively, and file system I/O accounts for a large portion in web browsers and multimedia players. Based on these observations, we suggested the adoption of NVM appropriately for given applications. In particular, for graph visualization, DB, and multimedia player applications, we suggest NVM to use as a swap, a journal, and a file system partitions, respectively. For DB applications such as OLTP, performances can be further improved if we use a small portion of NVM as a hot file partition.

ACKNOWLEDGMENT

This work was supported by the Basic Science Research program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4015750) and ICT R&D program of MSIP/IITP (R-20160904-004151).

REFERENCES

[1] E. Lee, D. Jin, K. Koh, and H. Bahn, "Is Buffer Cache Still Effective for High Speed PCM (Phase Change Memory) Storage?" Proc. IEEE Int'l Conf. Parallel and Distributed Systems (ICPADS), 2011.

[2] C. D. Wright, M. M. Aziz, M. Armand, S. Senkader, and W. Yu, "Can We Reach Tbit/sq.in. Storage Densities with Phase-Change Media?" Proc. European Phase Change and Ovonic s Symp. (EPCOS), 2006.

[3] F. Bedeschi et al., "A multi-level-cell bipolar-selected phase-change memory," Proc. Int'l Solid-State Circuits Conf. (ISSCC), 2008.

[4] A. M. Caulfield, A. De, J. Coburn, T. I. Mollov, R.K. Gupta, and S. Swanson, "Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories," Proc. IEEE/ACM Symp. Microarchitecture (Micro), pp.385-395, 2010

[5] A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson, "Onyx: a prototype phase change memory storage array," Proc. USENIX Conf. Hot topics in Storage and File systems (HotStorage), 2011.

[6] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, "Better I/O through byte-addressable, persistent memory," Proc. ACM Symp. Operating Systems Principles (SOSP), 2009.

[7] E. Lee, J. Jang, T. Kim, and H. Bahn, "On-demand Snapshot: An Efficient Versioning File System for Phase-Change Memory," IEEE Tran. Knowledge and Data Engineering, vol. 25, no. 12, pp.2841-2853, 2013.

[8] H. Wong, S. Raoux, S. Kim, J. Liang, J. Reifenberg, B. Rajendran, M. Asheghi, and K. Goodson, "Phase Change Memory," Proc. of the IEEE, vol.98, no.12, pp.2201-2227, 2010.

[9] B. Nale, R. Ramanujan, M. Swaminathan, and T. Thomas, "Memory Channel that Supports near Memory and Far Memory Access," PCT/US2011/054421, Intel Corporation, 2013.

[10] R. K. Ramanujan, R. Agarwal, and G. J. Hinton, "Apparatus and Method for Implementing a Multi-level Memory Hierarchy Having Different Operating Modes," US 20130268728 A1, Intel Corporation, 2013.

[11] R. F. Freitas, and W. W. Wilcke, "Storage-class memory: The next storage system technology," IBM J. Res. and Dev., vol.52, no.4, pp.439-447, 2008.

[12] M. Kryder, and C. S. Kim, "After hard drives: What comes next?," IEEE Tran. Magnetics, vol.45, no.10, pp.3406-3413, 2009.

Jisun Kim received the B.S degrees in the computer science and engineering from Hanshin University in 2011. She is currently a PhD candidate of computer science and engineering at Ewha Womans University, Korea. Her research interests include operating system, storage system, caching algorithms, system optimization and embedded systems.

Hyokung Bahn received the BS, MS, and PhD degrees in computer science from Seoul National University, in 1997, 1999, and 2002, respectively. He is currently a full professor of computer engineering at Ewha University, Korea. His research interests include operating systems, storage systems, embedded systems, and real-time systems. He received the Best Paper Awards at the USENIX Conference on File and Storage Technologies in 2013.