IEIE Transactions on Smart Processing and Computing

Evaluation of Various Workloads in Filebench Suitable for Phase-change Memory

Seungyong Lee[1], Hyokeun Lee[1], Hyuk-Jae Lee[1], and Hyun Kim[2]

[1] Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
{sylee, hklee, hyuk_jae_lee}@capp.snu.ac.kr
[2] Department of Electrical and Information Engineering and Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul, Korea hyunkim@seoultech.ac.kr

* Corresponding Author: Hyun Kim

Received October 23, 2020; Revised November 16, 2020; Accepted December 4, 2020; Published April 30, 2021

** Regular Paper*

** Extended from a Conference: Preliminary results of this paper were presented at the summer annual conference of IEIE2020. This paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

Abstract: Phase-Change Memory (PCM) is known as the next generation of memory thanks to its outstanding properties, such as fast speed, non-volatility, scalability, and low power consumption. Based on these characteristics, PCM can be used in larger expanded memory or faster storage compared to HDD or NAND flash memory. Therefore, various companies are trying to deploy PCM-based memory products. However, studies on deciding the target system of PCM are still insufficient, which is an obstacle to the commercialization of PCM. In this paper, a file system benchmark, *Filebench*, is evaluated with various operating options to find the most appropriate workload for PCM as storage. An experiment was conducted in a virtual system by mounting PCM with a PCM-aware file system. The results demonstrate that the PCM-based system performs up to 500 times better than traditional storage if executing workloads with a significant amount of write operations and synchronization operations. A number of applications were tested on various configurations of systems, and workload characteristics suitable for the PCM-based system are presented.

Keywords: Non-volatile memory, Phase-change RAM, File system, Storage

1. Introduction

DRAM is reaching its limit due to scalability and power consumption problems, and further progress is becoming increasingly difficult. Additionally, applications entailing tremendous memory resources, such as deep learning or big data management, are creating demand for new types of memory [1-3]. As a result, non-volatile memory (NVM) is emerging as a next-generation technology. In particular, phase-change memory (PCM) is a type of NVM that is expected to replace DRAM in the future as it has fast speed close to that of DRAM, high scalability, and low power consumption. PCM typically exhibits performance between that of DRAM and NAND flash memory, so it can be used as both main memory and storage.

However, it is problematic to apply PCM without considering its differences from conventional memory devices (DRAM, HDDs, and NAND flash memory). If it is used as main memory, the speed and bandwidth of PCM are lower than that of DRAM, so it is not enough to completely replace DRAM. Significant performance degradation occurs if PCM is used as storage because double-copy from PCM to DRAM occurs when the architecture is composed of a DRAM cache and block driver [4].

Furthermore, since PCM can be accessed in byte granularity, applying it directly to an existing file system has a performance limit in fully using the advantages of PCM. Because PCM has different characteristics from existing storage, it is impossible to apply prior knowledge about applications used in storage. This is one of the reasons why many vendors have difficulty finding the target applications of PCM and launching commercial products. Consequently, sufficient analysis of applications suitable for PCM is necessary.

Previous studies have focused on the device characteristics of PCM or algorithms to improve its performance [5-10]. Hence, few studies have analyzed the conditions of an operating environment suitable for PCM. There are many studies on PCM-optimized file systems in terms of storage that do not analyze the operating environment and appropriate targets [11-13]. Some studies use PCM in a variety of other ways, including hypervisors and databases, but have not yet been specifically targeted for PCM [14-16].

In response to these needs, the performance of PCM as storage was evaluated in this study by running different applications of *Filebench* to obtain workload characteristics that are suitable for PCM. PCM was emulated in a virtual environment and mounted with a direct access (DAX) file system. Therefore, the byte-addressability of PCM can be used, which is the main characteristic that results in differences from conventional devices. With the emulated PCM, workloads were evaluated with varying amounts of files, IO size, and numbers of threads of the workloads.

From this evaluation, the proper conditions for PCM were specified, and the differences between PCM and conventional storage (HDDs and SSDs) were determined. PCM outperforms conventional storage if executing workloads with write-intensive properties and significant synchronization operations, which results in up to 500× better performance. Based on these observations, an application suitable for PCM is proposed.

The rest of this paper is organized as follows. Section 2 presents the background of this research. Section 3 describes the proposed methodologies for workload evaluation. Section 4 shows the emulation environment and evaluation results. Finally, Section 5 concludes the study.

2. Background

2.1 Phase-change Memory

PCM stores information by alternating the phase of $Ge_2Sb_2Te_5$ (GST). The phase of the material is an amorphous state or crystalline state and determines the resistance. It can be changed by applying different temperatures to a cell, so PCM is considered as a representative NVM. It can be highly integrated since it has high scalability without introducing a large number of capacitors. Furthermore, it has byte-granularity and latency similar to DRAM, so it is expected to replace DRAM in the future.

2.2 Direct Access (DAX)

Page cache is widely used in conventional block device storage. It is a buffer between memory and storage when a system reads or writes files. The latency of storage can be hidden by caching data in page cache (DRAM). However, the use of page cache in a memory-like block device results in degraded performance due to unnecessary copies. Furthermore, the page cache has no persistency, so a system crash hurts the integrity of data in the page cache even if the storage has a non-volatility feature.

DAX allows an application to access storage directly by load/store commands. Therefore, a file system with DAX capability prevents duplication in the page cache by skipping it. Moreover, DAX provides better performance when applications with a number of small writes or synchronization operations are executed on the target system.

2.3 Filebench

Filebench is a file system and storage benchmark that is widely used in academia and industry [17]. Workload Model Language (WML) is a high-level language defined by *Filebench* that makes it easy to change the configuration of workloads, observe the performance of each operation, or even create new workloads in a few lines of code. In this study, *Filebench* was used to test workloads with various characteristics and obtain detailed results easily. Four representative pre-defined workloads were evaluated: Fileserver, Webserver, Webproxy, and Varmail. The features of each workload are as follows:

- Fileserver: Fileserver is a workload that simulates the operation of a file server and performs various file management commands, such as creating, writing, opening, reading, appending, and deleting files. Other workloads only write by appending, but Fileserver has writewholefile command and appendfile command, so it has a write-intensive feature.
- Webserver: Webserver is a workload that receives an HTTP request, reads a large amount of files, passes it to the client, and stores access records in a log file. It has a read-intensive feature and no directory operation.
- Webproxy: Webproxy acts as a normal web proxy server that receives requests from clients and seeks resources from other servers. Reading files is its main feature, and it has many directory operations like createfile and deletefile.
- Varmail: Varmail is a workload that functions as a mail server that receives mail, reads it, and synchronizes it after marking them read. The difference from other workloads is the fsync command, which is the synchronization command of *Filebench*.

3. System Description

Experiments were performed in a virtual environment after building a system. Fig. 1 shows an overview of the constructed system. Inside the virtual machine (guest), workloads read and write files to virtual storage. If the workload issues read or write commands, it reaches the storage through the file system.

Fig. 1(a) shows the file system layer, where ext4-DAX and ext4 are used for virtual PCM (vPCM) and virtual storage (vSTG), respectively. Because vPCM is mounted

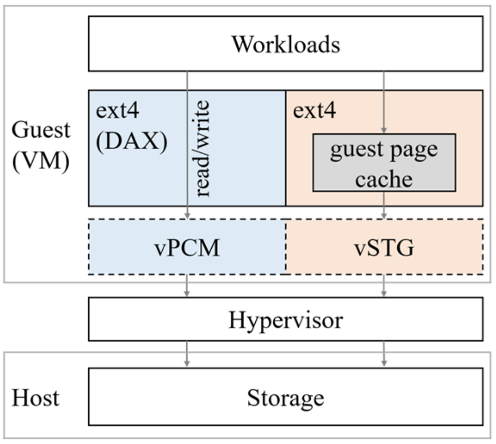


Fig. 1. Overview of simulation system (a) File system layer, (b) virtual storage layer.

with a DAX-enabled file system, there is no guest page cache in vPCM's file system layer. Therefore, applications can access the storage using load/store commands in byte granularity. Since the file system of vSTG has a guest page cache, read and write operations are executed in the guest page cache in typical cases.

Fig. 1(b) presents the virtual device layer of the guest. This layer acts as storage for the guest, which is actually emulated on a file in the host storage. Therefore, commands need to go through the hypervisor layer to access the actual storage. Commands to the guest storage enter the hypervisor layer, arrive at the host storage, and perform read/write operations on the file.

The host storage has a file where vPCM and vSTG are emulated. Consequently, the performance of workloads is affected by the speed of accessing the file. Therefore, the type of storage must be the same when comparing vPCM and vSTD with each other to find the benefits of using the DAX feature. This study uses HDDs, SSDs, and DRAMs as host storage to find the advantages of PCM over traditional storage.

The performance gap between vPCM and vSTG is mainly caused by the page cache. Because of the DAX feature of vPCM, read and write commands skip the guest page cache and reach the host storage through the hypervisor. A command to vPCM has some overhead since it always goes through the hypervisor layer and accesses the host storage. In contrast to vPCM, read and write commands to vSTG are buffered once in the guest page cache and then reach the storage after eviction.

If the guest page cache works well due to high locality, the vSTG can hide its low latency. On the other hand, when the fsync command is used, which flushes the page cache, vPCM's file system does not need additional work since it does not use the guest page cache. But the fsync command in vSTG actually conducts a flush, causing significant direct access to the storage, leading to much performance degradation.

Table 1. Summary of workload configuration

	Fileserver	Webserver	Webproxy	Varmail
# of files	100K	100K	100K	50K
File size	128KB	64KB	32KB	16KB
I/O size (R/W)	1MB/16KB	1MB/8KB	1MB/16KB	1MB/16KB
Directory width	20	20	1M	1M
Runtime	60s	60s	60s	60s
# of threads	50	50	50	50
R/W ratio	1:2	10:1	5:1	1:1

4. Evaluation

4.1 Simulation Environment

Experiments were done using a virtual machine with Linux on a QEMU emulator [18], which corresponds to the hypervisor in Fig. 1. For easy installation of the library, Fedora 28 (kernel version 4.18) was used. The virtual environment system uses an Intel i7-7800X 3.5-GHz core and 16 GB of DRAM. 64-GB PCM (vPCM in Fig. 1) was emulated on a file (storage in Fig. 1) that is located in each DRAM, HDD (WD20EZRZ), and SSD (TS512GSSD230S) according to the experiments. It is recognized in the virtual system using LIBNVDIMM [19], a sub-system that manages NVDIMM devices in Linux, and NDCTL, a library for the user-interface control of LIBNVDIMM [20]. The PCM and conventional storage are mounted with ext4-DAX and ext4, respectively. The host page cache is turned off in all experiments.

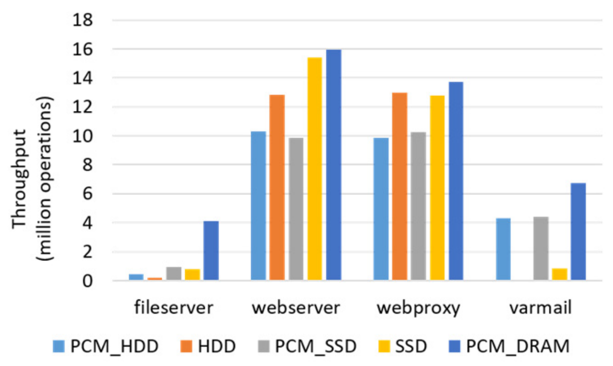
4.2 Workload Configuration

The four *Filebench* workloads mentioned in Section 2 were evaluated. The basic configuration of the workloads modified from previous studies is presented in Table 1 [11, 13]. The parameters are configured in a way that preserves the characteristics of each workload. It should be noted that the number of files of Varmail (50K) is half that of other workloads (100K) because Varmail originally has a small file set.

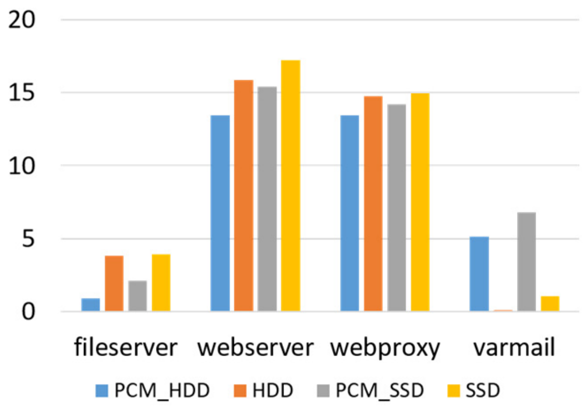
The experiments were performed by changing the amount of files, I/O size, and number of threads. The effect of the amount of files was tested by comparing the default workloads and the workloads with 10-times fewer files. The change in throughput and latency was observed while increasing the I/O size from 64 B to 16 KB and the number of threads from 1 to 32. Since there is a small variation in each run of a workload, all experiments were done five times.

4.3 Throughput of Default Configurations

Fig. 2(a) presents the *Filebench* throughput of the default setting with 5 different cases: PCM emulated on a file in an HDD (PCM-HDD), PCM emulated on a file in an SSD (PCM-SSD), and PCM emulated on a file in



(a) Default configuration



(b) Small file configuration

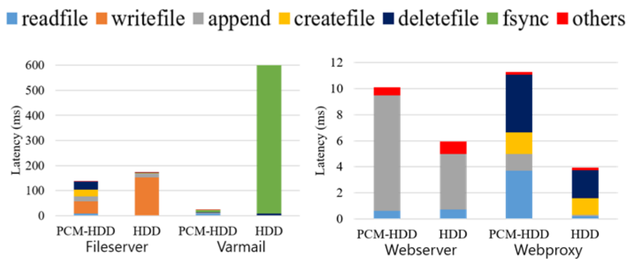
Fig. 2. Throughputs on each Filebench with various memory devices and configurations.

DRAM (PCM-DRAM), an HDD, and an SSD. The experimental results show that PCM-DRAM has the best performance in all workloads, so it can achieve the most similar performance to that of actual PCM devices.

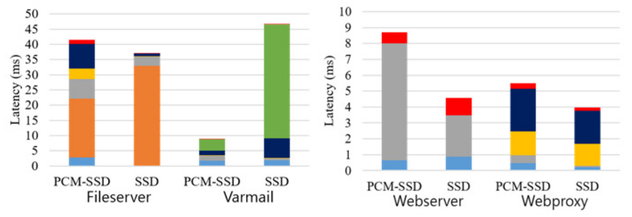
When comparing the other two PCMs (PCM-HDD and PCM-SSD) with conventional storage (HDD and SSD), the PCMs outperform the conventional storage in the write-intensive environments (Fileserver and Varmail) thanks to the DAX feature. In particular, Varmail shows a significant performance gap because there are many fsync commands that do not have the benefit of the guest page cache. On the other hand, in read-intensive Webserver and Webproxy, the HDD and SSD outperform PCMs because the hypervisor layer overhead becomes notable and the gain of the guest page cache almost reaches the performance of DRAM. From these results, it can be concluded that the DAX feature does not give a performance improvement in read-intensive environments and is suitable for environments with many writes and synchronization commands.

4.4 Latency Breakdown of Each Workload

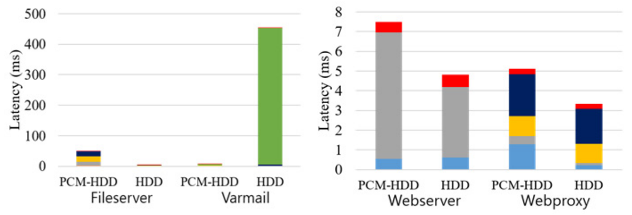
Fig. 3 shows a plot of the latency breakdown of *Filebench* workloads. Overall, the read-intensive workloads have shorter latency than the write-intensive workloads. Fig. 3(a) shows that in Fileserver, the write commands of HDD take 3 times longer than those of



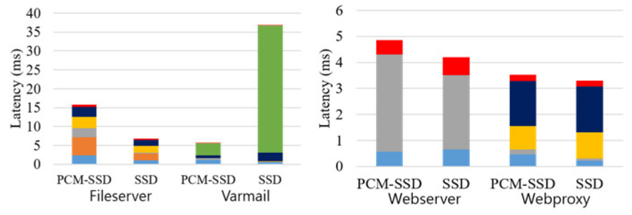
(a) PCM-HDD and HDD w/ default configuration



(b) PCM-SSD and SSD w/ default configuration



(c) PCM-HDD and HDD w/ small file configuration



(d) PCM-SSD and SSD w/ small file configuration

Fig. 3. Latency breakdown on each Filebench with various memory devices and configurations.

PCM-HDD and are responsible for the largest part of the latency. In Varmail, the fsync commands are responsible for a small portion of the total latency of PCM-HDD, but on HDDs, they cause a 200 times longer latency than that of PCM-HDD and are responsible for almost all the latency of the HDD.

Webserver and Webproxy show similar latency breakdown patterns on both the PCM-HDD and HDD, but the HDD shows better performance for append and readfile commands. When the storage is changed to the SSD, as shown in Fig. 3(b), the whole latency is remarkably reduced because the SSD latency of writefile in Fileserver and that of fsync in Varmail show a significant improvement. In the case of PCM-SSD, the latency of readfile in Webproxy is significantly reduced. From these results, it can be concluded that PCM shows high performance for the writefile and fsync commands, but not for the readfile commands, which can cause many transactions.

4.5 Experimental Results According to Various Operating Environments

Figs. 2(b), 3(c) and (d) show the experimental results with regard to the amount of files. Small amounts of files improve the performance on most workloads due to their high locality [11], as shown in Fig. 2(b). The change of HDD and SSD in Fileserver is especially notable. There is a significant improvement compared to the default configuration (Fig. 2(a)) because the small number of files can fit in the page cache, causing a decrease of latency from writefile commands.

On the other hand, as shown in Figs. 3(c) and (d), the latency of all commands is evenly reduced due to the high locality. This shows that PCM has no advantages of a small amount of files because conventional storage has a better throughput improvement in write-intensive environments due to small amounts of files and still shows higher throughput in read-intensive environments. Changing the directory width leads to similar results as changing the amount of files.

Fig. 4 shows the measured throughput with regard to the I/O size. The variation of latency can be inferred by taking the inverse of the throughput. In all devices, larger I/O size causes more pre-fetch effects and fewer transactions, resulting in better performance, but the throughput converges around 4K. Webserver, Webproxy, and Varmail present this tendency clearly. On Webserver, which has many read operations with high locality, convergence happens a bit late. This result shows that the I/O size of PCM should be set to 4K based on the convergence of the throughput.

Fig. 5 presents the average latency under a varying number of threads. It shows the latency on a log scale in order to show the correlation between the latency increase and the number of threads. Like in the previous experiments, the variation of throughput can be inferred by taking the inverse of the latency. For most workloads, the latency increases proportionally to the number of threads, which means increasing the number of threads does not help to improve the performance due to additional latency.

In Varmail, the slope of the graphs of HDD and SSD is very gradual because the guest page cache can be flushed simultaneously on HDDs or SSDs. Therefore, their latency is more tolerable to the growing number of threads compared to PCMs. However, even if this tolerance is present, it should be noted that the absolute value of the latency is significantly higher than that of PCM's latency. Therefore, it can be concluded that PCM has no advantages over conventional storage, depending on the number of threads.

4.6 Application Proposal

The experiments demonstrate that PCM performs better than traditional storage on workloads with the following characteristics: many synchronization commands, write-intensive commands, low locality, and 4-KB I/O size. Hence, applications that write a small-size unit in a large amount of data and perform frequent metadata updates are proposed as a target for PCM.

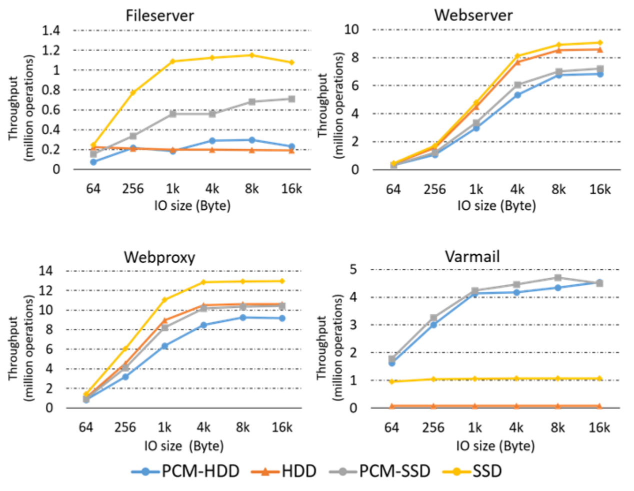


Fig. 4. Throughput of Filebench with varying I/O size.

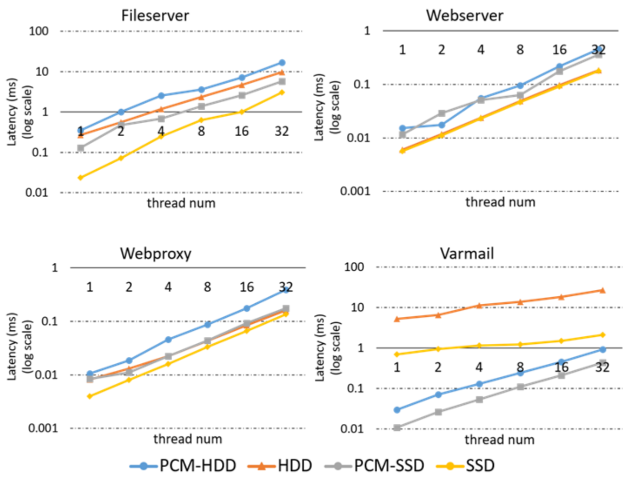


Fig. 5. Latency of Filebench with varying the number of threads.

5. Conclusion

In this study, *Filebench* workloads with various configurations were evaluated on a PCM-aware file system. The workload characteristics suitable for PCM were found by comparing the performance of each configuration. The simulation results showed that PCM performs well for workloads with a number of write or synchronization operations. Based on the observations, target properties suitable for PCM were proposed, which could be a great help in the development and use of PCM. The results are expected to contribute significantly to the commerciali-zation of PCM.

Acknowledgement

This paper was supported in part by the Technology Innovation Program (10080613, DRAM/PRAM heterogeneous memory architecture and controller IC design technology research and development) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea, and in part by a National Research Foundation of Korea

(NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1057530).

References

[1] B. Kim et al., "PCM: Precision-Controlled Memory System for Energy Efficient Deep Neural Network Training," in *Proc. 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1199-1204, Apr 2020. Article (CrossRef Link)

[2] D. T. Nguyen, N. H. Hung, H. Kim, and H.-J. Lee, "An Approximate Memory Architecture for Energy Saving in Deep Learning Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 67, no. 5, pp. 1588-1601, May 2020. Article (CrossRef Link)

[3] C. Lee and H. Lee, "Effective Parallelization of a High-Order Graph Matching Algorithm for GPU Execution," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 560-571, Feb. 2019. Article (CrossRef Link)

[4] M. Kim, I. Chang and H. Lee, "Segmented Tag Cache: A Novel Cache Organization for Reducing Dynamic Read Energy," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1546-1552, 2019. Article (CrossRef Link)

[5] L. Jiang, Y. Zhang and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *Proc. 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 216-227, 2014. Article (CrossRef Link)

[6] H. Lee, M. Kim, H. Kim, H. Kim and H. Lee, "Integration and Boost of a Read-Modify-Write Module in Phase Change Memory System," *IEEE Transactions on Computers*, vol. 68, no. 12, pp. 1772-1784, 1 Dec. 2019. Article (CrossRef Link)

[7] S. Kim, H. Jung, W. Shin, H. Lee, and H.-J. Lee, "HAD-TWL: Hot Address Detection-Based Wear Leveling for phase-change memory systems with low latency," *IEEE Computer Architecture Letters*, vol. 18.2, pp. 107-110, 2019. Article (CrossRef Link)

[8] R. Wang et al., "Decongest: Accelerating super-dense PCM under write disturbance by hot page remapping." *IEEE Computer Architecture Letters*, vol. 16.2, pp. 107-110, 2017. Article (CrossRef Link)

[9] H. Lee, H. Jung, H. Lee, and H. Kim, "Bit-width Reduction in Write Counters for Wear Leveling in a Phase-change Memory System." *IEIE Transactions on Smart Processing & Computing* 9.5 (2020): 413-419. Article (CrossRef Link)

[10] M. Kim, J. Lee, H. Kim, and H. Lee. "An On-Demand Scrubbing Solution for Read Disturbance Error in Phase-Change Memory." in *Proc. 2020 International Conference on Electronics, Information, and Communication (ICEIC),* pp.110-111, Jan. 2020. Article (CrossRef Link)

[11] J. Xu and S. Swanson, "NOVA: A log-structured file system for hybrid volatile/non-volatile main memories," in *Proc. 14th Usenix Conference on File and Storage Technologies*, pp.323–338, 2016. Article (CrossRef Link)

[12] J. Ou, J. Shu and Y. Lu, "A high performance file system for non-volatile main memory," in *Proc. Eleventh European Conference on Computer Systems*, no. 12, pp. 1-16, 2016. Article (CrossRef Link)

[13] M. Dong and H. Chen, "Soft updates made simple and fast on non-volatile memory," in *Proc. 2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pp.719-731, 2017. Article (CrossRef Link)

[14] L. Liang et al., "A case for virtualizing persistent memory," in *Proc. Seventh ACM Symposium on Cloud Computing (SOCC 2016)*, pp. 126-140, 2016. Article (CrossRef Link)

[15] N. U. Mustafa, A. Armejach, O. Ozturk, A. Cristal and O. S. Unsal, "Implications of non-volatile memory as primary storage for database management systems," in *Proc. 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pp. 164-171, 2016. Article (CrossRef Link)

[16] C. Wu, G. Zhang and K. Li, "Rethinking computer architectures and software systems for phase-change memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12.4, pp. 1-40, 2016. Article (CrossRef Link)

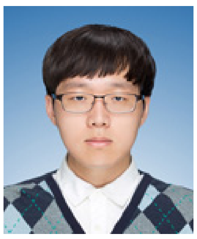
[17] V. Tarasov, E. Zadok and S. Shepler, "Filebench: A flexible framework for file system benchmarking," *login: The USENIX Magazine*, vol. 41.1, pp. 6-12, 2016. Article (CrossRef Link)

[18] F. Bellard, "QEMU, a fast and portable dynamic translator," in *Proc. USENIX Annual Technical Conference, FREENIX Track*, vol. 41, pp. 46, 2005. Article (CrossRef Link)

[19] LIBNVDIMM: Non Volatile Devices, Article (CrossRef Link)

[20] NDCTL, Article (CrossRef Link)

Seungyong Lee received a B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2018. He is currently working toward integrated M.S. and Ph.D. degrees in electrical and computer engineering at Seoul National University. His current research interests include non-volatile memory controller design, processing-in-memory, and computer architecture.



Hyokeun Lee received a B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently working toward integrated M.S. and Ph.D. degrees in electrical and computer engineering. His current research interests include nonvolatile memory controller design, hardware persistent models for non-volatile memory, and computer architecture.

Hyuk-Jae Lee received B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and a Ph.D. degree in electrical and computer Engineering from Purdue University, West Lafayette, IN, USA, in 1996. From 1998 to 2001, he was a Senior Component Design Engineer at the Server and Workstation Chipset Division, Intel Corporation, Hillsboro, OR, USA. From 1996 to 1998, he was a Faculty Member at the Department of Computer Science, Louisiana Tech University, Ruston, LS, USA. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, where he is a Professor. He is the Founder of Mamurian Design, Inc., Seoul, a fabless SoC design house for multimedia applications. His current research interests include computer architecture and SoC for multimedia applications.

Hyun Kim received B.S., M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was a BK Assistant Professor at the BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is an Assistant Professor. His current research interests include algorithms, computer architecture, memory, and SoC design for low-complexity multimedia applications, and deep neural networks.

Copyrights © 2021 The Institute of Electronics and Information Engineers