Characterizing a Real PCM Storage Device

Ju-Young Jung, Kelli Ireland, Jiannan Ouyang, Bruce Childers, Sangyeun Cho,
Rami Melhem, Daniel Mossé, Jun Yang[†], Youtao Zhang, and A. J. Camber[‡]
Computer Science Department
Electrical and Computer Engineering Department[†]
University of Pittsburgh
Micron Technology, Inc.[‡]

I. Motivation

It is anticipated that some or all of the roles assumed by conventional rotating hard disk drives (HDDs) will migrate to solid-state storage devices (SSDs) and emerging persistent RAM storage devices (PSDs) [1]–[4]. SSDs are typically built with solid-state NAND flash memory and have already been deployed in data centers because of their performance and energy merits. Increasingly more researchers are exploring the potential of the SSD as HDD replacement or complement. While emerging persistent RAMs like PCM (Phase Change Memory) are touted as having more desirable physical characteristics than NAND flash memory, to date, there has been little work evaluating actual PSDs empirically [5], [6].

In this paper, we characterize the capabilities and peculiarities of the PSD as SSD alternative with a PSD prototype system built with 90nm PCM chips. We first identify which design parameters of the PSD may affect storage system performance and describe our measurement process. Next, we present the performance of the PSD interacting with the most recent Linux OS and provide some insights on the differences between the PSD and a SSD when serving I/O requests. Our results suggest that the different PSD and SSD characteristics should be taken into account to design an efficient storage system.

II. Evaluation Approach

A. Experimental Environment

In this work, we examine a 12GB PSD prototype from Micron. This device employs an array of PCM chips as main storage medium and uses the PCIe (PCI Express) $\times 8$ bus for host interfacing. It supports a 4KB (or larger) block size. The prototype is comprised of three daughter boards mounted on a main controller board, and each daughter board has its own FPGA controller. Our PSD host system is a Dell OPTIFLEX 980 desktop computer running a Linux 2.6.35 kernel. For our study we run the *fio* file system benchmark workload [7] with an ext3 file system under various parameter settings. During repeated experiments, the block size and the target file size are fixed to 4 KB and 1 GB respectively. We compared the PSD with a SATA-attached Samsung MCB4E50G5MXP SSD.

B. Design Parameters

There are many design parameters which may affect storage system performance and thus must be considered in the process of PSD evaluation. Broadly, they are classified into two major categories: *workload-level* and *system-level parameters*, listed in the following tables.

TABLE I. Workload-level Parameters.

Parameters	Descriptions
Block size	Unit data transfer size affects I/O throughput, and PCM's byte-addressability may affect optimal block size
File size	Aggregate file sizes (working set size) as well as individual file size impact storage performance
I/O depth	Performance is sensitive to the maximum number of outstanding requests
Read/write mix	The mixture ratio of read/write requests attracts special interest because of unbalanced read/write performance
Disk fullness	The effective storage capacity gradually decreases. This incurs performance different from FOB (fresh out-of-box)
Request pattern	I/O requests arrivals can be either sporadic or bursting. This presents insights on the temporal behaviors of PSD.
Multi I/O threads	The number of I/O threads raises the per-thread I/O performance and the fairness problem of I/O service.
Access region	I/O requests condensed to only small portions of PSD have different spatial behaviors from even distribution.

There is another set of performance parameters that should be considered at the system-level and they are related to OS activities and other system components.

TABLE II. System-level Parameters.

Parameters	Descriptions
OS I/O software stack latency	Accessing PSD through traditional I/O software stack will give us baseline performance. Default HDD-optimized or noop I/O schedulers in block I/O layer may account for more portions of overall performance than the past. File object caching optimizations in page cache layer may allow additional performance improvements.
Boot time	System boot-up performance is another system-level concern and is dominated by the latency to copy kernel images from storage to main memory.
File system	Various file systems may exhibit performance differences. Identifying the strengths and weaknesses of them may allow for file system design improvements. Also, it may guide file system designers to design new file system for PSD.

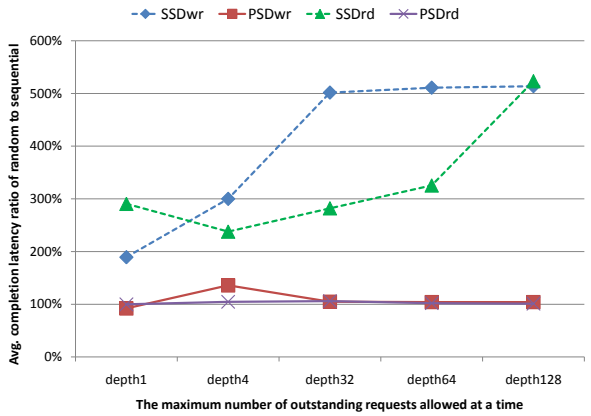

Fig. 1. Sensitivity to randomness: This illustrates the performance of random read and write operations normalized to that of sequential read or write operations over both SSD and PSD; the completion latency is defined by the time taken to complete a given request since it have been issued to the I/O queue.

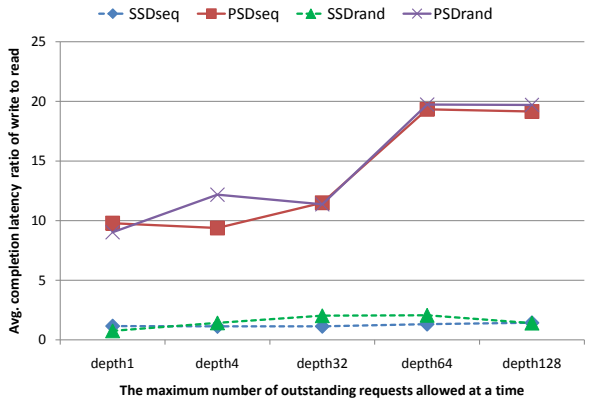

Fig. 2. Degree of performance unbalance between read and write. Write performance has been normalized to read performance and measured by the completion latency.

III. Preliminary Results

In this section, we present results that capture the performance difference between PSD and SSD. We measured the performance of Linux's asynchronous I/O operation under the ext3 file system. We always begin with a warm-up period and run the experiments five times to eliminate noises.

Fig. 1 shows the average latency (measured in μsec) of random write and read operations for both PSD and SSD when normalized to its sequential counterparts. In particular, we observe the sensitivity when changing the number of outstanding I/O requests allowable. While the PSD maintains almost identical performance regardless of sequential or random read/write requests, the SSD is much more sensitive to random I/O requests, showing performance degradations of up to +500% for both reads and writes.

Meanwhile, both the SSD and PSD have a property in common—read and write performance at the level of memory cell are unbalanced. Fig. 2 shows how this asymmetry affects the average completion latency of write operations normalized to read operation latency, for both devices. While the SSD has 1.5 times slower write performance than read

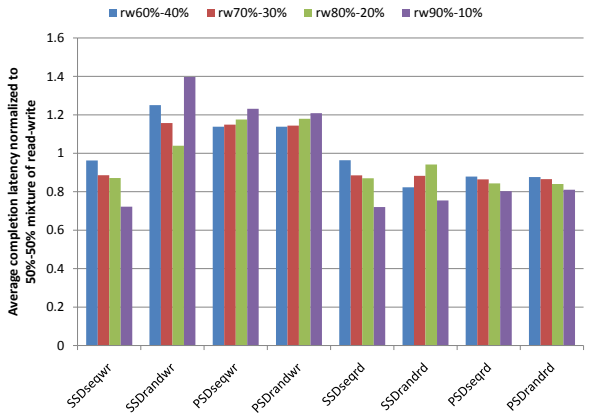

Fig. 3. Sensitivity to requests' read/write mix: the result is normalized to the default mix of R/W = 50%/50%.

for random operation and 1.2 times for sequential operation on average, the PSD has a large performance gap between read and write operation; the gap can be as large as 19 times for random and even sequential operation. This implies that PSD designers must address the asymmetric read/write performance problem more carefully.

Lastly, Fig. 3 shows the sensitivity of the studied PSD and SSD devices to the read/write request mix. We measure the performance with 50% disk fullness and the I/O depth of 64 entries (best PSD performance for the *fio* workload). Again, the PSD reveals different behaviors from SSD. For example, as the write ratio decreases, sequential write performance of the SSD improves while that of the PSD continues to degrade.

IV. Current Status

At the time of writing this paper, the majority of workload-level parameters listed in TABLE I have been examined. We expect that the upcoming exploration of the remaining workload-level and system-level parameters will be equally insightful. For example, we will measure not only the low-level performance by accessing the PSD without a file system in a raw device mode, but also the improvement of software latency when the conventional OS software stack is revamped. Additionally, the comparison between the PSD and a PCIe-attached SSD will be interesting as both share the same interface type.

References

[1] M. H. Kryder et al., "After Hard Drives—What Comes Next?," *IEEE Transactions on Magnetics*, 45(10): 3406–3413, 2009.
[2] J. Condit et al., "Better I/O through byte-addressable, persistent memory," *SOSP'09*, pp 133–146, 2009.
[3] J. C. Mogul et al., "Operating System Support for NVM+DRAM Hybrid Main Memory," *HotOS XII*, May 18–20, 2009.
[4] E. Seppanenand et al., "High Performance Solid State Storage Under Linux," *MSST'10*, pp 1-12, May 2010.
[5] A. Caufield et al., "Moneta: A High-performance Storage Array Architecture for Next-generation, Non-volatile Memories," *Micro'10*, December 2010.
[6] J. D. Davis et al., "FRP: A Nonvolatile Memory Research Platform Targeting NAND Flash," *WISH'09*, March 2009.
[7] J. Axboe, FIO - Flexible I/O Tester, `http://freshmeat.net/projects/fio/`.