

Efficient management of PCM-based swap storage

Yunjoo Park¹, Sungyoung Ahn², and Hyokyung Bahn^{1a)}

¹ Department of Computer Engineering, Ewha University,
52, Ewhayodae-gil, Seodaemun-Gu, Seoul, Korea

² Memory Division of Samsung Electronics Co., Ltd.,
16 Banwol-Dong, Hwasung-city, Gyeonggi-Do, Korea

a) bahn@ewha.ac.kr

Abstract: This letter explores the performance of PCM-based swap storage and presents three management techniques. First, we attach PCM on DIMM slots to eliminate block I/O overhead and remove the context switching process while handling page faults. Second, we reduce the page size and turn off the read-ahead option to consider the high performance characteristics of PCM. Third, we decrease the DRAM memory size to save energy consumption without sacrificing the performance.

Keywords: phase-change memory, non-volatile memory, swap storage, page replacement, write references

Classification: Storage technology

References

- [1] E. Lee, H. Bahn and S. Noh: Proc. USENIX Conf. File and Storage Technol. (2013) 73.
- [2] E. Lee, K. Koh and H. Bahn: IEICE Electron. Express **11** (2014) 20140520. DOI:10.1587/elex.11.20140520
- [3] B. Lee, E. Ipek, O. Mutlu and D. Burger: Commun. ACM **53** (2010) 99. DOI:10.1145/1785414.1785441
- [4] S. Lee, H. Bahn and S. Noh: IEEE Trans. Comput. **63** (2014) 2187. DOI:10.1109/TC.2013.98
- [5] E. Lee, H. Bahn, S. Yoo and S. Noh: Proc. IEEE MASCOTS Conf. (2014) 405. DOI:10.1109/MASCOTS.2014.56
- [6] E. Lee, J. E. Jang, T. Kim and H. Bahn: IEEE Trans. Knowl. Data Eng. **25** (2013) 2841. DOI:10.1109/TKDE.2013.35
- [7] J. Gim, T. Hwang, Y. Won and K. Kant: ACM Trans. Storage **9** (2013) 39. DOI:10.1145/2631922
- [8] E. Lee, Y. Kim and H. Bahn: IEICE Trans. Inf. & Syst. **E97-D** (2014) 323. DOI:10.1587/transinf.E97.D.323
- [9] N. Nethercote and J. Seward: Electron. Notes Theor. Comput. Sci. **89** (2003) 44. DOI:10.1016/S1571-0661(04)81042-9
- [10] H. Yoon et al.: SAFARI Technical Report No. 2013-002 (2013).

1 Introduction

Due to the recent advances in non-volatile memory technologies such as PCM (phase-change memory) and STT-MRAM (spin torque transfer magnetic RAM), a new memory hierarchy of computer systems based on these emerging media is expected to appear soon [1, 2, 3]. Specifically, PCM has been considered as a replacement of DRAM due to its low-power consumption and high density features [4]. However, the access time of PCM is slow (about 2–5× read, 8–50× write that of DRAM) to be a main memory medium of computer systems [5].

To cope with this situation, we adopt PCM as high-speed swap storage and discuss how such systems can be managed efficiently. In traditional HDD-based swap storage, a large amount of adjacent data are loaded together when a page fault occurs since the seek time of HDD is very large [6]. To improve CPU utilization, a process that incurs a page fault becomes blocked and the CPU is switched to other process while handling the page fault. However, this may not be efficient for PCM swap storage that is sufficiently fast and does not have seek time [7, 8].

This letter presents three management techniques for systems using PCM swap storage. First, we reduce the page fault handling time by attaching PCM on DIMM slots and eliminate the software stack overhead of block I/O; we also improve performance by removing the context switching process while handling page faults. Second, we show that reducing the page size and turning off the read-ahead option is effective under the PCM swap storage where the page fault handling time is very small. Third, we show that the performance is not degraded even with small DRAM memory under the PCM swap storage; this leads to the reduction of DRAM's energy consumption significantly compared to HDD swap systems.

2 Empirical evaluations

We capture memory access traces from four Linux applications: freecell, gqview, kghostview, and xmms. To extract these traces, we developed a trace collector by modifying the Cachegrind tool of Valgrind 3.2.3 [9]. The characteristics of the captured traces are described in Table I.

Using these traces, we perform simulation experiments. We developed a set of memory and swap management simulators that have the ability of configuring various system parameters such as the page size, the read-ahead window size, the page replacement algorithm, the total memory capacity, and the swap I/O latency. The software architecture of the simulator consists of five modules: the MMU simulation module, the page fault handling module, the page replacement module, the swap management module, and the energy simulation module. When simulation begins, our simulator parses each line of the trace and invokes the MMU simulation module. The MMU simulation module references the page table and handles the request within DRAM or calls the page fault handling module if the requested page is not in memory. The page fault handling module invokes the swap management module that returns the swap I/O latency for the given configurations. It also invokes the page replacement module if there is no free page in memory.

In the page replacement module, the CLOCK algorithm, which is widely used in virtual memory systems, is adopted as a baseline algorithm [4]. We assume that

PCM is put on DIMM slots and there is no context switching while handling page faults. The access time of a PCM device consists of a static time component needed for each access and a time component proportional to the request size [4, 6, 10]. Suppose that the proportional time component of a write and a read operation is T_{p_write} and T_{p_read} , respectively, and the static time component of a read and a write is T_{i_read} and T_{i_write} , respectively. When the total number of read and write operations is N_{read} and N_{write} , respectively, and the page size is $SIZE_p$, then the total elapsed time T for accessing PCM swap storage is

$$T = N_{read}(T_{i_read} + T_{p_read} * SIZE_p) + N_{write}(T_{i_write} + T_{p_write} * SIZE_p).$$

We also implemented an energy simulation module that measures the energy consumption of memory during simulations. The energy simulation module calculates the energy consumption of the memory system based on the static and active power consumptions. Static power consumption includes the leakage power and the refresh power. The leakage power is power consumed even when memory is idle. DRAM cells store data in small capacitors that lose their charge over time and must be recharged. This process is called refresh. Regardless of read and write operations, DRAM consumes considerable refresh power to sustain refresh cycles to retain its data. Active power consumption refers to the energy dissipated when data is being read and written. Then, total power consumption P_{total} is calculated as

$$P_{total} = P_{static} + P_{active}$$

where

$$P_{static} = Unit_static_power (W/GB) * memory_size (GB) \text{ and}$$

$$P_{active} = (N_{read} * E_{read}(J) + N_{write} * E_{write}(J))$$

$$/(N_{read} * L_{read}(ns) + N_{write} * L_{write}(ns)).$$

$Unit_static_power$ is the static power per capacity, and N_{read} and N_{write} are the number of reads and writes, respectively. E_{read} and E_{write} refer to the energy required for a read and a write operation, respectively. L_{read} and L_{write} are the average latency of a read and a write operation, respectively. Table II lists the parameters used in our experiments.

Fig. 1 shows the total elapsed time of each workload as the page size is varied. For a comparison purpose, we also show the performance of HDD swap systems. For readers' convenience, we use different scales for PCM and HDD in the y-axis. The results show that PCM-swap performs better as the page size becomes smaller whereas the best performance of HDD-swap is obtained when the page size is 4 KB or more. This indicates that the page size of 4 KB, which is commonly used in modern systems, will not be efficient for PCM swap systems.

Table I. Characteristics of the traces used in the experiments.

Workload	Footprint (KB)	Ratio of ops. (r:w)	Total Accesses
freecell	10,080	7.16 : 1	490,175
gqview	7,430	1 : 1.30	610,685
kghostview	17,390	13.93 : 1	1,546,135
xmms	8,050	1 : 5.13	1,168,939

Table II. Parameters used in our experiments

	E_{read}	E_{write}	L_{read}	L_{write}	$Unit_static_power$
DRAM	0.1 (nJ/bit)	0.1 (nJ/bit)	50 (ns)	50 (ns)	1 (W/GB)
PCM	0.2 (nJ/bit)	1.0 (nJ/bit)	50 (ns)	500 (ns)	0.1 (W/GB)

However, reducing the page size is not a simple issue since the number of page table entries should be increased as the page size shrinks. This would eventually increase the memory requirement for the page table. However, it is likely that the total memory capacity can be reduced when a small page size is used because a variety of contents can be accommodated even with a small DRAM capacity, and the page fault handling process is fast enough under PCM-swap. Therefore, we argue that the space overhead of the page table can be relieved by reducing the total DRAM capacity of the system. Specifically, if we use the page size of 512 B instead of 4 KB, the number of pages in the system is increased by 8 times. However, this does not increase the memory requirement for the page table if we reduce the total DRAM capacity of the system to 1/8. We show that our system consisting of small memory (1/10 of the original memory capacity) and fast swap storage exhibits a competitive performance and is also energy-efficient although it does not increase the page table overhead.

Fig. 2 shows the total elapsed time of each workload executed under PCM-swap systems as the DRAM capacity is varied from 5% to 100% of the memory footprint. For each configuration, we also vary the page size. We use 256 B as the minimum page size because the page size cannot be smaller than the block size managed in the last-level cache. As shown in the figure, the performance is not degraded even with the 5–10% DRAM capacity when a small page size (e.g., 256 B) is used. This is because only the necessary part to execute the program is loaded into memory. Note that HDD has a significant portion of size-independent time components to access data, whereas PCM does not do so, and thus the transfer size is critical to PCM performances. As we eliminate block I/O and context switching overheads from the page fault handling process, reducing the transfer size is important to minimize the total cost of a page fault.

Based on these preliminary experiments, we set the DRAM capacity to 10% of the memory footprint hereinafter since a reasonably good performance can be obtained with such DRAM capacity when we use a small page size. Fig. 3

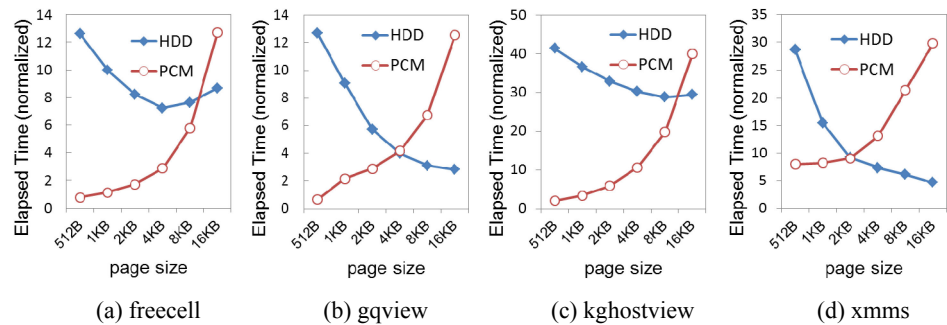


Fig. 1. HDD and PCM swap performances as a function of the page size.

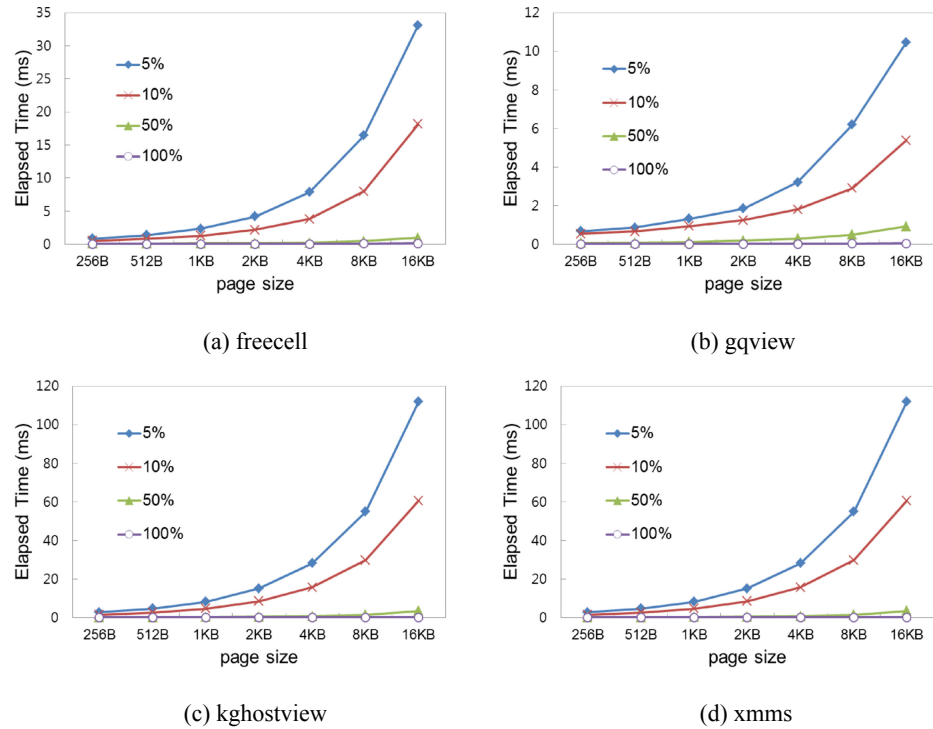


Fig. 2. Total elapsed time as a function of the page size.

compares the performance of the “baseline system” that uses the 4 KB page and the “small page system” that uses the 256 B page. Fig. 3 also shows the performance of PCM-swap when the read-ahead (RA) option is turned off.

Modern operating systems use the read-ahead technique that loads several adjacent pages as well as the requested page itself when a page fault occurs. This is effective in HDD-swap to minimize the head movement as storage accesses frequently exhibit sequential behavior. However, as shown in Fig. 3, that is not the case in PCM-swap because the cost of loading unnecessary pages through read-ahead is relatively expensive in PCM.

Since a write operation in PCM is significantly slower than a read operation, it is necessary to predict future write accesses and maintain pages likely to be written again in memory. To this end, we devise a new page replacement algorithm called W-CLOCK, by simply changing the original CLOCK algorithm such that the *dirty bit* is used to capture the recency of references instead of the *reference bit*.

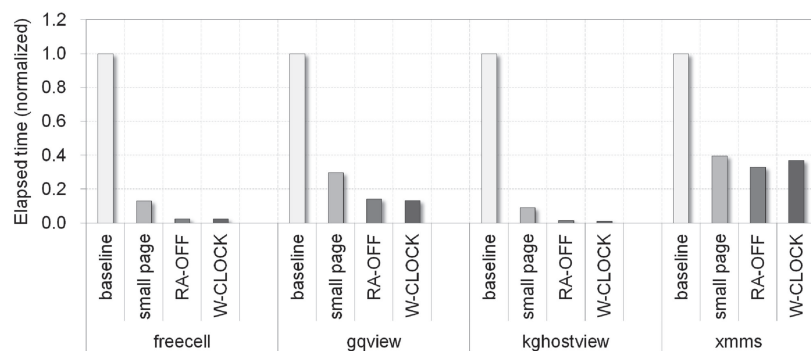


Fig. 3. PCM swap performance with three management techniques.

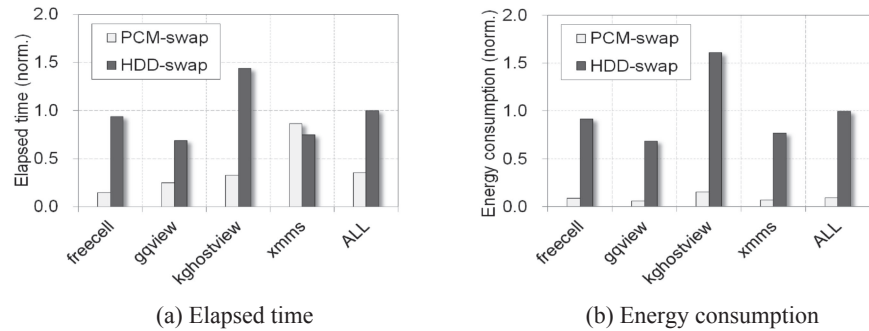


Fig. 4. Comparison of total elapsed time and energy consumption.

Specifically, W-CLOCK checks the dirty bit of the page to which the clock-hand points, and if it is 1, W-CLOCK clears the dirty bit and backs it up in the kernel page structure. By so doing, kernel still knows that it is a dirty page that should be written back to storage if selected as an eviction victim. If a page with the dirty bit of 0 is found, that page is then replaced. As shown in Fig. 3, adopting W-CLOCK further improves the performance of PCM-swap in most cases. Though W-CLOCK may not be effective in some workloads like xmms, it reduces the number of expensive writes to PCM by estimating the re-reference likelihood of write operations well.

Finally, Fig. 4 shows the total elapsed time and the energy consumption of “PCM-swap” under the 10% DRAM size in comparison with the “disk-based conventional swap” under the 100% DRAM size. For PCM-swap, we adopt all the aforementioned management techniques. As shown in the figure, PCM-swap exhibits 64.1% better performance than HDD-swap on average but consumes 90.2% less energy. The only case PCM-swap performs worse than HDD-swap is xmms, which is excessively write-intensive. Nevertheless, the energy consumption of PCM-swap is still less than HDD-swap in this case. This is because all DRAM cells need consistent refresh operations to retain data regardless of any read/write requests. This result indicates that the new swap architecture of “small memory – fast swap” will be effective in reducing the energy consumption of future computer systems.

3 Conclusion

This letter explored the performance of PCM-based swap storage systems and showed that reducing the page size and turning off the read-ahead option are effective in PCM-swap. We also showed that the performance is not degraded but the DRAM’s energy consumption is significantly reduced when we use only a small amount of DRAM memory. We expect that our result will lead to the transition of legacy swap systems of “large memory – slow swap” to a new paradigm of “small memory – fast swap.”

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0028825).