International Journal of Internet, Broadcasting and Communication Vol.14 No.1 188-193 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.1.188

IJIBC 22-1-23

Energy-Efficient Last-Level Cache Management for PCM Memory Systems

Hyokyung Bahn

Professor, Department of Computer Engineering, Ewha University, Korea
bahn@ewha.ac.kr

Abstract

The energy efficiency of memory systems is an important task in designing future computer systems as memory capacity continues to increase to accommodate the growing big data. In this article, we present an energy-efficient last-level cache management policy for future mobile systems. The proposed policy makes use of low-power PCM (phase-change memory) as the main memory medium, and reduces the amount of data written to PCM, thereby saving memory energy consumptions. To do so, the policy keeps track of the modified cache lines within each cache block, and replaces the last-level cache block that incurs the smallest PCM writing upon cache replacement requests. Also, the policy considers the access bit of cache blocks along with the cache line modifications in order not to degrade the cache hit ratio. Simulation experiments using SPEC benchmarks show that the proposed policy reduces the power consumption of PCM memory by 22.7% on average without degrading performances.

Keywords: *Mobile system, Last-level cache, Phase-change memory, Main memory, Energy-efficiency*

1. Introduction

Recently, using NVM (non-volatile memory) as the memory layer of systems is attracting attention to reduce the power consumption in various computing environments including PC [1], mobile [2], and industrial systems [3]. Specifically, PCM (Phase-Change Memory) is discussed as a main memory of future computer systems as it consumes less power than DRAM and has a non-volatile characteristic [1, 4]. However, it is difficult to replace DRAM in general-purpose computer systems as PCM is slower than DRAM. In mobile IoT devices, however, since the amount of computation is not large and power-saving is more important than the performance of memory [5], PCM can be a strong candidate for the main memory medium [1, 4]. Meanwhile, PCM has a special characteristic in write operations such that the write latency is 7-10 times slower than that of DRAM, and the number of write operations allowed per each cell is limited to 10^7-10^8 times [4].

In this article, we propose a last-level cache management technique for mobile systems where PCM is used as a main memory medium. Cache management has been studied widely in various system layers to buffer slow memory or storage systems [6, 7]. As the capacity of the cache layer is limited, in order to insert new data items in cache when there is no available space, some items in the cache should be discarded. In this

Manuscript Received: January. 13, 2022 / Revised: January. 17, 2022 / Accepted: January. 21, 2022
Corresponding Author: bahn@ewha.ac.kr
Tel: +82-2-3277-2368, Fax: +82-2-3277-2306
Professor, Department of Computer Engineering, Ewha University,Korea

situation, a replacement algorithm is necessary to determine the victim block in the cache, where a cache block is the unit of managing data items in cache memory. Traditionally, cache replacement algorithms aim to increase the hit ratio by replacing cache blocks that are unlikely to be used again in the near future. Meanwhile, cache blocks residing in the cache memory can be classified into two categories: unmodified blocks, in which only a read operation occurs after entering the cache memory, and modified blocks, in which a write operation occurs at least once. Unmodified blocks can simply be erased when they are replaced from cache memory, whereas modified blocks should be reflected to main memory before being erased.

We aim to reduce the write traffic to main memory by classifying the state of modified blocks into multi-levels based on the amount of writing to be generated to main memory, and then replacing the cache block with a small amount of writing when a free cache block is necessary. Specifically, we subdivide a cache block into a certain number of cache lines, and when a write operation occurs on a cache block, we keep track of the cache line the write occurred on; when a cache block needs to be replaced, we evict a block with the smallest number of modified cache lines preferentially. This is because PCM we use as the main memory has a slow write operation and allows a limited number of possible writes, so we need to delay or avoid the write operation as much as possible.

Meanwhile, the main mission of the replacement algorithm is to maximize the cache hit ratio, which is the ratio of data accessed directly from the cache memory not requiring main memory accesses. Thus, in addition to reducing the amount of writing to PCM main memory, it is also important to increase the cache hit ratio. To do so, we keep the traditional principle of caching, that limits the blocks of not used recently as victim candidates, and of those, select a cache block with the smallest number of modified lines as the victim cache block. This has the effect of reducing the amount of data written to PCM without degrading the performance of the cache memory. Simulation experiments with SPEC benchmarks show that the proposed policy reduces the write traffic to PCM significantly without degrading the cache hit ratio, leading to reduced power consumption.

2. An Energy-Efficient Last-Level Cache Management Policy

In our architecture, SRAM is used as cache memory similar to conventional architectures, and PCM is used as main memory instead of DRAM. Figure 1 depicts the proposed system architecture. Our policy is adopted in the last-level cache memory on top of the PCM main memory. In CPU, there are the first-level and the second-level cache memories, and the last-level cache memory transfers data from and to the second-level

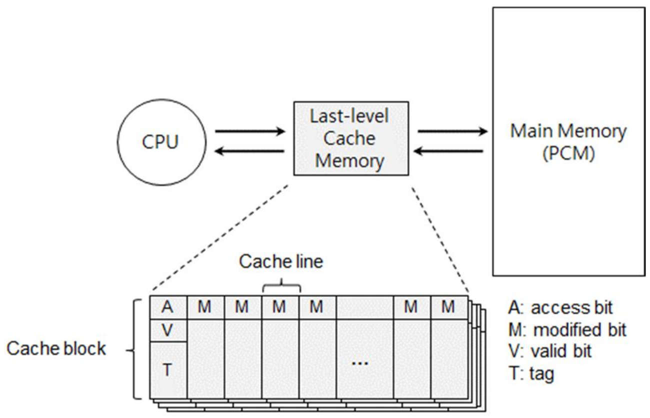


Figure 1. Cache memory architecture of the proposed policy.

cache memory in a cache line granularity, whereas it transfers data from and to the PCM main memory in a cache block unit. In our policy, we make use of the 16-way set associative cache in the last-level cache memory, implying that there are 16 different locations on the cache memory that each cache block can enter. The sizes of a cache block and a cache line are set to 4 kilobytes and to 256 bytes, respectively. Thus, a cache block consists of 16 cache lines.

For each cache block, our policy makes use of an access bit and a valid bit in order to monitor whether the block is recently used and maintains valid data, respectively. A cache block also requires a modified bit to ensure whether it has been changed after entering the cache so that modifications can be flushed to main memory when the cache block is evicted. Our policy maintains a modified bit for each cache line to see the amount of writing when a cache block is selected as the target of replacement. The modified bit is set to 1 when a dirty cache line is flushed from the second-level cache memory. Note that the dirty bit of 1 in the last-level cache means that the cache line should be flushed to PCM main memory when the cache block it belongs to is evicted from the last-level cache memory.

Our policy chooses a victim block according to the state of modified bits and the access bit of each cache block to reduce the amount of data written to PCM main memory and improve the cache hit ratio at the same time. When a cache block is accessed, our policy sets the access bit of the block to 1 and it also sets the modified bit of the accessed cache line to 1 when the operation is a write. The access bits of all cache blocks in the last-level cache memory are periodically reset to 0 as is done by the CLOCK algorithm [4]. The access bit of each cache block indicates whether the block has recently been accessed or not. So, those blocks likely to be re-accessed in the near future and those that are not can be distinguished with this access bit. To improve the cache hit ratio, our policy restricts victim blocks for replacement to those blocks with the access bit of 0. In order to reduce the amount of writing to PCM as well, our policy checks the modified bits of cache blocks with their access bit of 0, and selects the block with the smallest number of modified cache lines as the eviction victim. This allows for blocks accessed recently to remain in the last-level cache, whereas reducing the amount of writing to PCM memory by discarding a cache block that does not incur much writing to PCM.

3. Simulation Experiments

To assess the effectiveness of the proposed policy, we perform simulation experiments for the proposed policy in comparison with the well-known caching policy CLOCK. CLOCK is a replacement policy widely adopted in modern cache memory systems, which evicts a block with its access bit 0 among the cache blocks; when the modified bit of the victim block is set to 1, CLOCK reflects the block to main memory before eviction. The baseline configuration of our experimental platform consists of a dual core processor of 2.4 gigahertz and the last-level cache memory of 16-way set associative, whose size can be varied from 512 kilobytes to 4 megabytes.

The cache block size of the last-level cache is set to 4 kilobytes, and the cache line size is 256 bytes. For the first-level cache memory, the size of the instruction cache is 16 kilobytes and the size of the data cache is 8 kilobytes and we use the direct mapping of 32-byte cache lines. For the second-level cache memory, the 8-way set associative cache of 32-byte cache lines with the total capacity of 256 kilobytes are used. For PCM main memory, the read and write latencies are set to 20 nanoseconds and 100 nanoseconds, respectively, and read and write energies are set to 0.2 nanojoules per bit and 1.0 nanojoules per bit, respectively. The static power consumption of the PCM memory is set to 0.1 watts per gigabytes. For performance evaluations, we use four SPEC benchmarks, namely gamess, gromacs, lbm, and sphinx3. The memory access traces used in our simulations are the last-level cache memory access traces captured by making use of the Cachegrind tool of Valgrind toolset [8].

Figure 2 depicts the amount of data written to PCM memory for the proposed policy in comparison with the original CLOCK policy as the size of the last-level cache is varied from 512 kilobytes to 4 megabytes. Note that we plot the result of our policy normalized to that of the CLOCK. As the figure shows, our policy reduces the amount of writing to PCM significantly compared to CLOCK for all cache sizes and workloads. Specifically, the reduced write traffic to PCM by making use of our policy is 27.3% on average and up to 42.8% compared to CLOCK. Figures 3 depicts the average memory access time of the proposed policy compared to CLOCK. As the figure shows, the two policies exhibit almost identical results regardless of the cache size and workloads although our policy reduces the amount of writing to PCM significantly. This is because our policy evicts a cache block with the same condition in terms of the access bit when compared to the original CLOCK algorithm. In other words, the two policies discard a cache block only if its access bit is 0. Figure 4 depicts the power consumption in main memory when using our policy in comparison with CLOCK. As we see in the figure, the proposed policy reduces the power consumption significantly compared to CLOCK. Specifically, the saved power by adopting our policy is 22.7% on average and up to 35.7%. This is because PCM consumes relatively large power for write operations compared to read, and the proposed policy suppresses PCM write operations as much as possible.

4. Conclusion

Energy efficiency of main memory and cache memory systems is becoming increasingly important in designing future computer systems as the size of in-memory data continues to grow. Low-power memory media such as PCM and STT-MRAM (spin-transfer torque magnetic random access memory) are expected to be strong candidates for the main memory medium of future mobile embedded systems as they consume less power than DRAM.

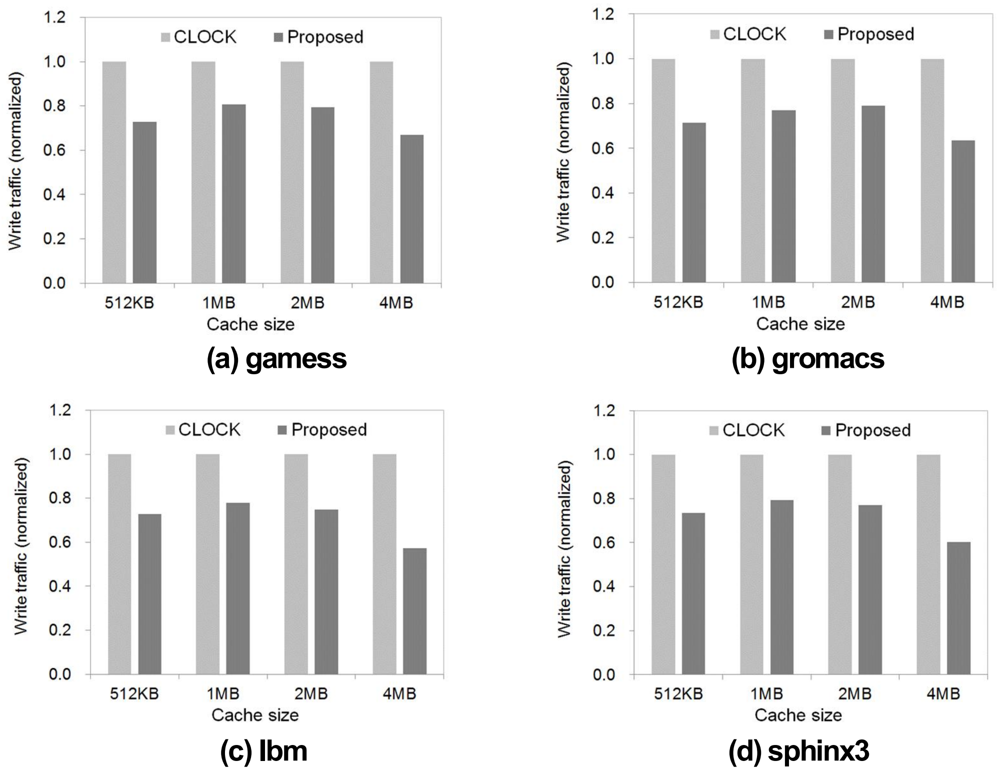
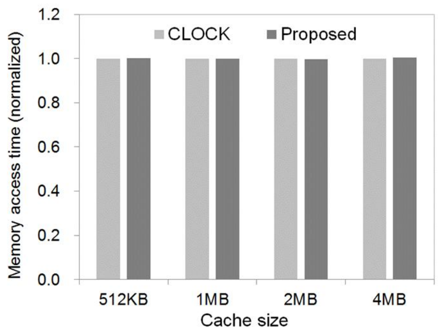
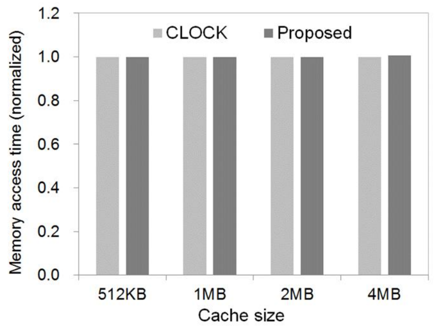


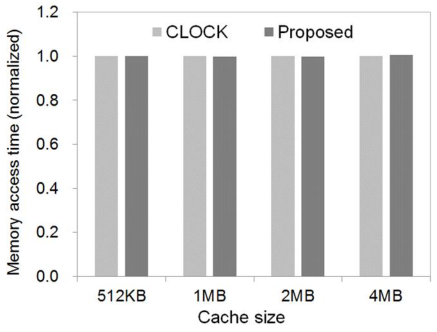
Figure 2. Comparison of the amount of writing to PCM memory.



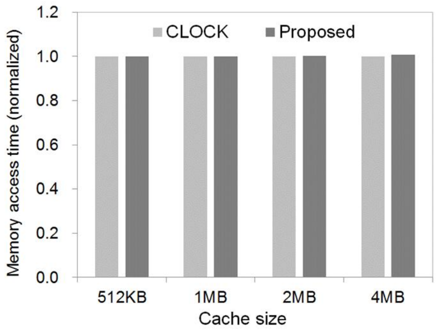
(a) gamess



(b) gromacs

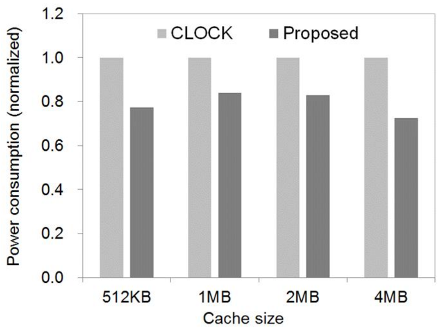


(c) lbm

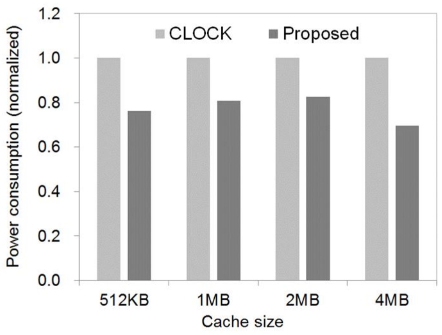


(d) sphinx3

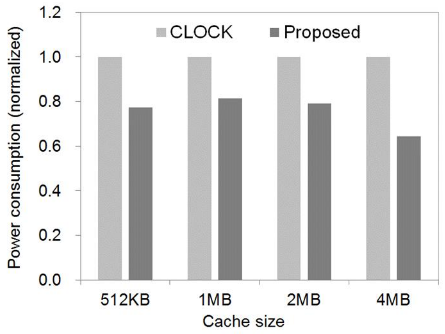
Figure 3. Comparison of the memory access time.



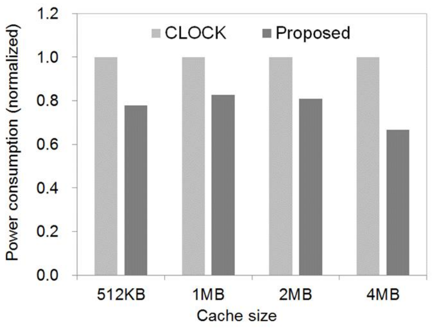
(a) gamess



(b) gromacs



(c) lbm



(d) sphinx3

Figure 4. Comparison of the power consumption.

In this article, we proposed a replacement policy for the last-level cache of energy-efficient mobile embedded systems. The proposed policy makes use of PCM as the main memory medium instead of DRAM, and aims at reducing the amount of writing to PCM by evicting cache blocks with the least number of modified cache lines preferentially among those cache blocks not used recently. Simulation experiments showed that the proposed policy reduces the amount of data written to PCM by 26% on average and up to 52% compared to the well-acknowledged CLOCK algorithm without performance degradations.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1009275) and also by the ICT R&D program of MSIT/IITP (2020-0-00121, development of data improvement and dataset correction technology based on data quality assessment).

References

[1] S. Lin and S. Wang, "Thermal-constrained Memory Management for Three-Dimensional DRAM-PCM Memory with Deep Neural Network Applications," Microprocessors and Microsystems, vol. 89, article 104444, 2022.
DOI: https://doi.org/10.1016/j.micpro.2022.104444

[2] J. Kim and H. Bahn, "Analysis of smartphone I/O characteristics—Toward efficient swap in a smartphone," IEEE Access, vol. 7, pp. 129930-129941, 2019.
DOI: https://doi.org/10.1109/ACCESS.2019.2937852

[3] S. Yoo, Y. Jo, and H. Bahn, "Integrated Scheduling of Real-time and Interactive Tasks for Configurable Industrial Systems," IEEE Transactions on Industrial Informatics, vol. 18, no. 1, pp. 631-641, 2022.
DOI: https://doi.org/10.1109/TII.2021.3067714

[4] S. Lee, H. Bahn, and S. Noh, "CLOCK-DWF: A Write-History-Aware Page Replacement Algorithm for Hybrid PCM and DRAM Memory Architectures," IEEE Transactions on Computers, vol. 63, no. 9, pp. 2187-2200, 2014.
DOI: https://doi.org/10.1109/TC.2013.98

[5] E. Lee and H. Bahn, "Electricity Usage Scheduling in Smart Building Environments Using Smart Devices," The Scientific World Journal, vol. 2013, article 468097, pp.1-11, 2013.
DOI: https://doi.org/10.1155/2013/468097

[6] O. Kwon, H. Bahn, and K Koh, "Popularity and prefix aware interval caching for multimedia streaming servers," Proceedings of the IEEE CIT Conference, pp. 555-560, 2008.
DOI: http://doi.org/10.1109/CIT.2008.4594735

[7] T. Kim and H. Bahn, "Implementation of the storage manager for an IPTV set-top box," IEEE Transactions on Consumer Electronics, vol. 54, no. 4, pp. 1770-1775, 2008.
DOI: http://doi.org/10.1109/TCE.2008.4711233

[8] N. Nethercote and J. Seward, "Valgrind: A Program Supervision Framework," Electronic Notes in Theoretical Computer Science, vol. 89, no. 2, pp. 44-66, 2003.
DOI: https://doi.org/10.1016/S1571-0661(04)81042-9