# Comparison of Page Caching Algorithms for Next Generation Mobile Computers with Hybrid Main Memory*

Soohyun Yang and Yeonseung Ryu

Department of Computer Engineering, Myongji University
Yongin, Gyeonggi-do, Korea
sh871201@naver.com, ysryu@mju.ac.kr

**Abstract.** As various next-generation non-volatile memories such as PRAM have been developed, it is expected that the next generation mobile computers will adopt these new memory technologies in near future. In this paper, we present some experiment results of page caching algorithms which consider DRAM/PRAM hybrid main memory and flash memory based storages. We have developed a trace-driven simulator to compare the performance in terms of several important performance metrics such as cache hit ratio.

**Keywords**: Page Caching, Mobile Computers, Flash Memory Storages, DRAM/PRAM Hybrid Main Memory

## 1 Introduction

Recently a number of mobile computers such as smart phone and tablet PC have been widely used. In most mobile computers, NAND flash based storages have been commonly adopted because of its superiority in fast access speeds and low power consumption. As various non-volatile memories such as PRAM (Phase change RAM), and MRAM (Magnetic RAM) have been developed as next generation memory technologies, it is expected that the next generation mobile computers will adopt these new memory technologies in near future. Among these non-volatile memories, PRAM is rapidly becoming a promising candidate for large scale main memory because of its high density and low power consumption. Though PRAM has attractive features, the write access latency of PRAM is not comparable to that of DRAM. Also, PRAM has a worn-out problem caused by limited write endurance. Since the write operations on PRAM significantly affect the performance of system, it should be carefully handled.

For several decades, DRAM has been used as the main memories of computer systems. However, recent studies have shown that DRAM-based main memory spends a significant portion of the total system power and the total system cost with the

* This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0021897).

increasing size of the memory system. In order to tackle the energy dissipation in DRAM-based main memory, some recent studies introduced PRAM-based main memory organization [1] and DRAM/PRAM hybrid main memory organization [2, 3]. Therefore, it is highly expected that the architecture of next generation mobile computers will adopt hybrid main memory and flash memory based storages in near future.

Most operating system (OS) including Linux employs a buffer cache mechanism to enhance the performance that is limited by slow secondary storage. For the past decades, buffer cache schemes have been implemented for DRAM-based main memory and hard disk based secondary storage. Recently, there have been some buffer cache schemes that consider erase limitation of flash memory storages [4-6]. Also, a few buffer cache schemes have been introduced to consider DRAM/PRAM hybrid main memory [7].

In this paper, we study legacy page caching algorithms considering non-volatile memories and present experiment results on their performance in terms of the cache hit ratio, the number of write counts on PRAM and the number of erase counts on flash memory. We also experiment the performance of a new page caching scheme called CF-BPLRU (Clean-first Block Padding LRU). The CF-BPLRU adopts advantages from two legacy page caching schemes: CFLRU and BPLRU. We show that the CF-BPLRU outperforms other schemes with regard to the erase counts on flash memory. The rest of this paper is organized as follows. In Section 2, we survey the legacy page caching algorithms for flash memory storages and PRAM-based main memory. Section 3 presents the experimental results. Finally, Section 4 concludes the paper.

## 2 Page Caching Algorithms for Non-volatile Memories 2.1

### Flash-aware page caching schemes

As the flash memory based storage devices have been widely used, page cache schemes have also been studied to consider the ‘erase-before-write’ characteristic of flash memory. A flash memory is organized in terms of blocks, where each block is of a fixed number of pages. The flash-aware page cache schemes can be classified into two categories: page-level and block-level schemes.

In [4], a page-level scheme called CFLRU (Clean first LRU) was proposed. CFLRU maintains the page list by LRU order and divides the page list into two regions, namely the working region and clean-first region. In order to reduce the write cost, CFLRU first evicts clean pages in the clean-first region by the LRU order, and if there are no clean pages in the clean-first region, it evicts dirty pages by their LRU order.

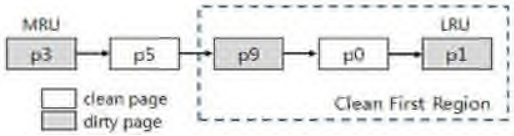



**Fig. 1**. Buffer cache structure of CFLRU.

For example, in Fig. 1, CFLRU selects the page p0 because it is the first clean page by the LRU order in the clean-first region. CFLRU can reduce the number of write and erase operations by delaying the flush of dirty pages in the page cache.

In [5, 6], block-level schemes called FAB (Flash Aware Buffer management) and BPLRU (Block Padding LRU) were proposed, which consider the block merge cost in the log block FTL schemes. Fig. 2 shows the buffer cache structure of block-level schemes. Block-level schemes maintain a LRU list based on the flash memory block. The LRU list is composed of block headers. Each block header manages buffers of member pages which are loaded from flash memory. When a page *p* of block *b* in the flash memory is first referenced, they allocate a new buffer and store page *p* in the allocated buffer. If the block header for block *b* does not exist, a new block header is allocated and placed at the MRU position of LRU list. Then, the buffer of page *p* is attached to the header of block *b*. Whenever a page in the buffer cache is referenced, all pages in the same block are moved to the MRU position.

When buffer cache is full, BPLRU scheme evicts all the pages of a victim block but it simply selects the victim block at the LRU position. In addition, it writes a whole block into a log block by the in-place scheme using the page padding technique. In page padding procedure, BPLRU reads some pages that are not in the victim block, and writes all pages in the block sequentially. The page padding may perform unnecessary reads and writes, but it is effective because it can change an expensive full merge to an efficient switch merge. In log-block FTL, all writes must be performed to log blocks, and the log blocks are merged with data blocks later. When a log block is written sequentially from the first page to the last page, it can simply replace the associated data block with a switch merge operation. If a victim block flushed by BPLRU is full, then a switch merge will be performed in the log-block FTL. In BPLRU, therefore, all log blocks can be merged by the switch merge, which results in decreasing the number erase operations.

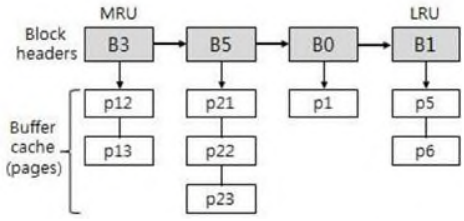



**Fig. 2**. Buffer cache structure of BPLRU.

In this paper, we propose a new page caching scheme, called CF-BPLRU, which is based on BPLRU and CFLRU. The proposed CF-BPLRU is almost the same as BPLRU. However, CF-BPLRU maintains the clean-first region in the LRU list like CFLRU (see Fig. 3). If all free buffers are used up, the CF-BPLRU selects a victim block from the clean-first region of the LRU list and flushes all pages of the victim block. Like BPLRU, in order to reduce the erase counts on flash memory, the CF-BPLRU reads some pages that are not in the victim block, and writes all pages in the block sequentially. The CF-BPLRU defines a *window* as *n* blocks from the LRU

position in the list. The size of window is defined as *n*. In Fig 4, for example, window size is 2.

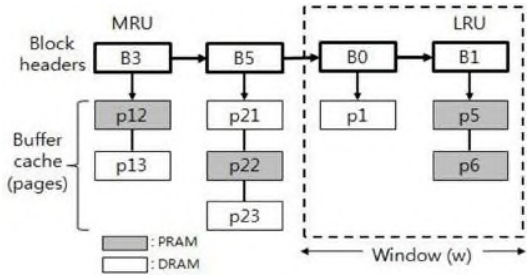



Fig. 3. Buffer cache structure of CF-

BPLRU. 2.2 PRAM-aware page caching schemes

For the DRAM/PRAM hybrid main memory, a migration-based cache scheme (we call it MIG) was proposed that predicts the page access pattern and migrates pages to DRAM or PRAM according to the access pattern [7]. For prediction of the access pattern, it calculates the weighting values of each page at every request and maintains 4 types of monitoring queues (see Fig. 4). The proposed scheme migrates a page when its weighting value is over or under predefined thresholds. However, it does not consider secondary storage characteristics like flash memory.

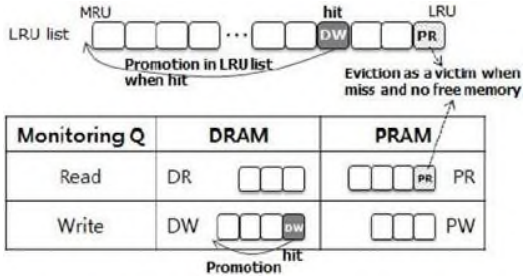



Fig. 4. Buffer cache structure of

MIG. 3 Simulation Results

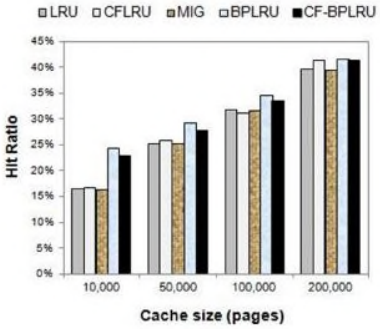
In this work, we have developed a trace-driven simulator to examine the performance of various page caching algorithms: LRU, CFLRU, MIG, BPLRU and CF-BPLRU.

We assume that the hybrid main memory consists of DRAM and PRAM, which are divided by a memory address. The memory which has the low memory address is DRAM and the high section is allocated to PRAM. When allocating a new buffer, we try to allocate it from the low section (i.e., we prefer DRAM buffer to PRAM buffer). We also assume that medium of storage device is flash memory. The flash memory

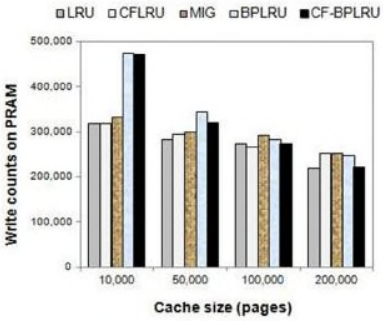
model used in the simulation was the Samsung 16GB NAND flash memory [8]. The page size is 4 KB and the number of pages in a block is 64. We implement BAST scheme as an FTL scheme of flash memory because it is a representative and basic log block scheme. In BAST scheme, 100 log blocks were used.

For the workload for mobile computers, we extracted disk I/O traces from notebook PC, running several applications, such as document editors, web browsers, media player and games. The read/write ratio of workload is 67%/33%. In the case of CFLRU and CF-BPLRU, we set the window size to 30% of maximum capacity of buffer cache as recommended in [4].

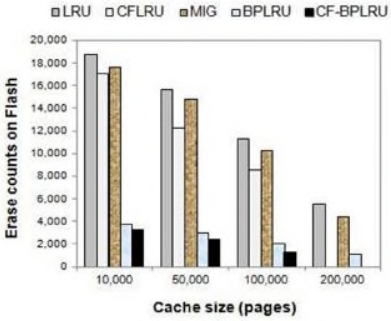
Fig. 5 shows experiment results. According to Fig. 5(a), the cache hit ratios of block-level schemes are greater than page-level schemes. In particular, when the cache size is 10,000, block-level schemes show worst performance. According to Fig. 5(b), the write counts on PRAM of the page-level schemes are less than other two block-level schemes. Fig. 5(c) shows that the block-level schemes are much better than other two page-level schemes in terms of erase count on flash memory. The reason is that the block-level schemes perform page padding to reduce merge overhead when it evicts victim block. Further, the CF-BPLRU can reduce the erase counts more because it tries to evict clean blocks.




(a) Cache hit ratio




(b) Write counts on PRAM




(c) Erase counts on flash memory

Fig. 5. Evaluation results.

## 4 Conclusion

In this paper, we have studied legacy representative page caching algorithms considering non-volatile memories and proposed a new page caching scheme called CF-BPLRU (Clean-first Block Padding LRU). In order to examine the performance characteristics, we have implemented trace-driven simulators and compared the performance in terms of the cache hit ratio, the number of write counts on PRAM and the number of erase counts on flash memory. We showed that page-level schemes outperform block-level schemes in terms of the cache hit ratio and write counts on PRAM. However, block-level schemes are much better than page-level schemes in terms of erase counts on flash memory. For the future works, we will study novel page caching algorithms as compromise between page-level scheme and block-level scheme.

## References


1. M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in *Proc. of International Symposium on Computer Architecture*, 2009.
2. G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A Hybrid PRAM and DRAM Main Memory System," in *Proc. of Design Automation Conference*, 2009.
3. H. Park, S. Yoo, and S. Lee, "Power Management of Hybrid DRAM/PRAM-based Main Memory," in *Proc. of Design Automation Conference*, pp. 59-64, 2011.
4. S. Park, D. Jung, J. Kang, J. Kim, and J. Lee. "CFLRU: A Replacement Algorithm for Flash Memory", in *Proc. of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pp. 234–241, 2006.
5. H. Jo, J. Kang, S. Park, and J. Lee, "FAB: Flash Aware Buffer Management Policy for Portable Media Players," *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, pp. 485-493, 2006.
6. H. Kim and S. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," in *Proc. of 6th USENIX Conference on File and Storage Technologies (FAST)*, pp. 239-252, 2008.
7. H. Seok, Y. Park, K.W. Park, and K.H. Park, "Efficient Page Caching Algorithm with Prediction and Migration for a Hybrid Main Memory," *Applied Computing Review*, Vol. 11, No. 4, 2012.
8. Samsung Electronics, K9XXG08UXM.1G x 8 Bit/2G x 8 bit NAND Flash Memory
9. J. Kim, J. Kim, S. Noh, S. Min, and Y. Cho, "A Space-efficient Flash Translation Layer for Compactflash Systems," *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, 2002.
10. Y. Ryu, "SAT: Switchable Address Translation for Flash Memory Storages," in *Proc. of IEEE Computer Software and Applications Conference (COMPSAC)*, Jul. 2010.