A Buffer Management Scheme for Mobile Computers with Hybrid Main Memory and Flash Memory Storages

Yeonseung Ryu

Department of Computer Engineering, Myongji University
Nam-dong, Yongin, Gyeonggido, Korea
E-mail: ysryu@mju.ac.kr

Abstract

Recently DRAM and PRAM hybrid main memory organization has been studied in order to address the high levels of energy dissipation in DRAM based main memory. It is expected that this new memory architecture will be used soon in mobile computers which use NAND Flash memory based storages. In such computers, legacy operating system functionalities like file system and memory system should be modified in order efficiently to manage heterogeneous memory organization. In this paper, we study a new buffer cache scheme which considers DRAM/PRAM hybrid main memory and flash memory based storages. The goal of proposed buffer cache scheme is to minimize the number of write operations on PRAM and the number of erase operations on flash memory while maintaining the cache hit ratio. In order to evaluate proposed scheme, we performed trace-driven simulation.

Keywords: *Buffer Cache, Buffer Replacement, Flash Memory, DRAM/PRAM Hybrid Main Memory*

1. Introduction

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 increasing size of the memory system. Fortunately, various non-volatile memories such as PRAM (Phase change RAM), FRAM (Ferroelectric RAM) and MRAM (Magnetic RAM) have been developed as a next generation memory technologies. Among these non-volatile memories, PRAM is rapidly becoming promising candidates for large scale main memory because of their high density and low power consumption. In order to tackle the energy dissipation in DRAM-based main memory, some studies introduced PRAM-based main memory organization [1] and DRAM/PRAM hybrid main memory organization [2, 3].

Figure 1 illustrates the system configuration considered in this paper. Our system uses DRAM and PRAM hybrid main memory and uses flash memory as storage device. Modern operating system (OS) supports a buffer cache mechanism to enhance the performance that is limited by slow secondary storage. When OS services a read request, it copies the data from storage to the buffer cache in the main memory and serves the next read operations from the faster main memory. Similarly, when OS services a write request, it stores data to the buffer cache and later flushes several data together to the storage. If the flash memory is used as storage device, OS usually employs a software layer called flash translation layer (FTL) [5-7]. An FTL receives read and write requests from the file system and maps a logical address to a physical address in the NAND flash. Recently, there have been studies on buffer cache management scheme that use flash memory as secondary storage. However, there are few

researches on buffer cache schemes that consider both hybrid main memory and flash memory-based storage device.

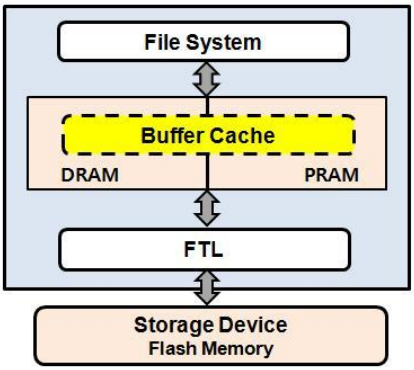


Figure 1. System Configuration

In this paper, we propose a new buffer cache management scheme called PABL (PRAM aware Block-based LRU) for hybrid main memory and flash storage devices. Proposed PABL scheme tries to minimize both the number of write operations on PRAM and the number of erase operations on flash memory. When PABL allocates a buffer to accommodate requested data, it allocates DRAM or PRAM according to the type of request. If the request type is read, the PABL tries to allocate buffer from PRAM. Otherwise, it tries to allocated buffer from DRAM. During the buffer replacement procedure, PABL considers merge operations performed in FTL in order to minimize the number of erase operations. Trace-driven simulation results show that PABL outperforms the legacy buffer cache schemes like LRU.

The rest of this paper is organized as follows. Section 2 gives an overview of the non-volatile memory, flash translation layer and the previous flash aware buffer cache management schemes. Section 3 explains the design and operation of the proposed scheme. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2. Background and Previous Works

2.1. Non-volatile Memories

A NAND flash memory is organized in terms of *blocks*, where each block is of a fixed number of *pages*. A block is the smallest unit of erase operation, while reads and writes are handled by pages [4]. Flash memory cannot be written over existing data unless erased in advance. The number of times an erasure unit can be erased is limited. The erase operation can only be performed on a full block and is slow that usually decreases system performance.

A PRAM cell uses a special material, called phase change material, to represent a bit. PRAM density is expected to be much greater than that of DRAM (about four times). Further, because the phase of the material does not change after power-off, PRAM has negligible leakage energy regardless of the size of the memory. 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.

2.2. Flash Translation Layer (FTL)

FTL is a kind of device driver in OS (see Figure 1). Most FTL schemes use a log block mechanism for storing updates [6, 7]. A log block scheme, called block associative sector translation (BAST), was proposed in [6]. In the BAST scheme, flash memory blocks are divided into data blocks and log blocks. Data blocks represent the ordinary storage space and log blocks are used for storing updates. When an update request arrives, the FTL writes the new data temporarily in the log block, thereby invalidating the corresponding data in the data block. Whenever the log block becomes full or the free log blocks are exhausted, garbage collection is performed in order to reclaim the log block and the corresponding data block. During the garbage collection, the valid data from the log block and the corresponding data block should be copied into an empty data block. This is called a *merge operation*. After the merge operation, two erase operations need to be performed in order to empty the log block and the old data block. When the data block is updated sequentially starting from the first page to the last page, the FTL can apply a simple *switch merge*, which requires only one erase operation and no copy operations. That is, the FTL erases the data block filled with invalid pages and switches the log block into a data block. The former merge operation is called a *full merge* as compared to the switch merge.

2.3. Previous Buffer Cache Schemes

In [8], CFLRU (Clean first LRU) was proposed to exploit the asymmetric performance of flash memory read and write operations. CFLRU maintains the page list of buffer cache 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. CFLRU can reduce the number of write and erase operations by delaying the flush of dirty pages in the buffer cache.

In [11, 12], block-level replacement 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 like BAST. These block-level replacement schemes maintain the block list by LRU order. When 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, FAB scheme searches a victim block from the LRU position which has the largest number of pages in the buffer cache. Then, all the pages of the selected block are passed to the FTL to flush into the flash memory. BPLRU scheme also evicts all the pages of a victim block like FAB, 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. Therefore, all log blocks can be merged by the switch merge.

These legacy schemes only focused on reducing the number of write operations to flash memory and did not address the write problem of PRAM.

3. Proposed Scheme

In this paper, we propose a new block-level buffer management scheme called PABL (PRAM aware Block-based LRU) for hybrid main memory and flash memory storage devices. Proposed PABL scheme tries to minimize both the number of write operations on PRAM and the number of erase operations on flash memory.

The PABL scheme maintains a LRU list based on the block-level as shown in Figure 2. The LRU list consists of block headers. Each block header manages buffers of affiliated pages which are loaded from flash memory. When a page *p* of block *b* in the flash memory is first referenced, the PABL allocates a new buffer and stores page *p* in the allocated buffer. If the block header for block *b* does not exist, the PABL allocates a new block header and attaches the buffer of page *p* to the header of block *b*. Further, *b* is placed at the MRU position of LRU list. Whenever a page in the buffer cache is referenced, all pages in the same block are moved to the MRU position.

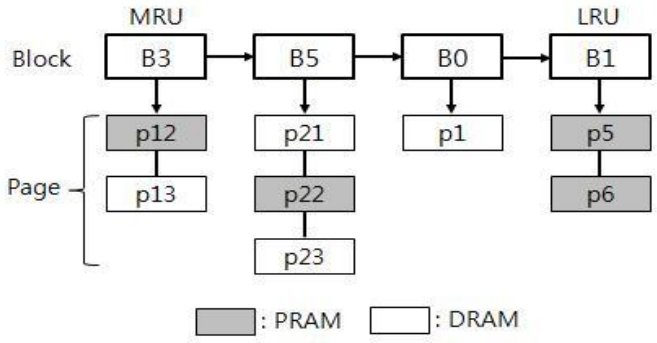


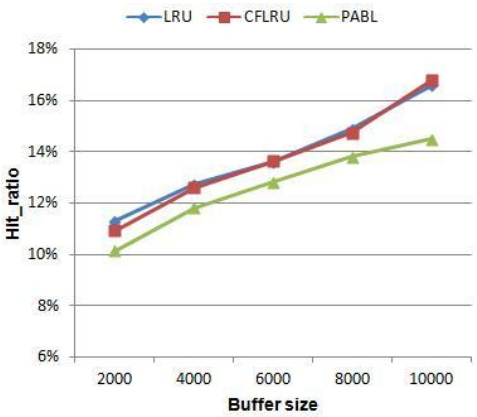
Figure 2. Buffer Cache Structure of PABL.

When PABL allocates a new buffer, it allocates DRAM or PRAM according to the type of request. If the request type is read, the PABL tries to allocate buffer from PRAM. Otherwise, it tries to allocated buffer from DRAM. If all free buffers are used up, the PABL must perform buffer replacement procedure. During the buffer replacement procedure, the PABL will select a victim block from the LRU position. If the victim block contains dirty pages, then the PABL performs page padding technique when it flushes the victim block to the flash memory in order to consider block merge overhead. If the victim block contains no dirty pages, the PABL can make the pages of the victim block free without write operations to the flash memory (no flushing).

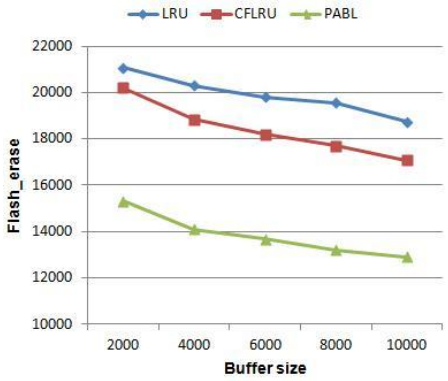
4. Simulation Results

In order to evaluate the proposed scheme, we compared PABL with two of the existing schemes using simulation: LRU and CFLRU. 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. In case of previous schemes, we assume that they allocates buffer from DRAM and PRAM alternately. 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 [4]. 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. We ignored the map block update cost in the BAST implementation.

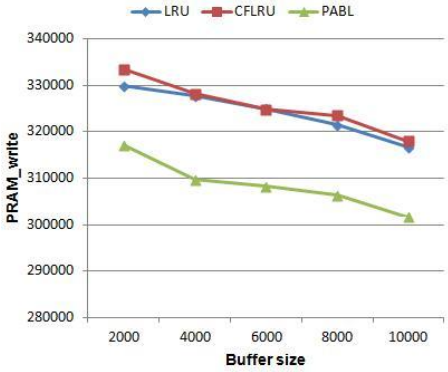
For the workload for our experiments, we extracted disk I/O traces (we call it PC) from Microsoft Windows XP-based notebook PC, running several applications, such as document editors, web browsers, media player and games. In workload PC, read/write



(a) Hit ratio



(b) Erase count on Flash



(c) Write count on PRAM

Figure 3. Performance Comparison Varying Buffer Cache Size

ratio is "67%/33%. We measured the hit ratio, the required number write operations on PRAM and the required number of erase operations on flash memory while varying the buffer cache size from 4 to 20MB. As the hit ratio of CFLRU is affected by the window size, in this experiment, we set it to 30% of maximum capacity of buffer cache.

Figure 3 shows experiment results. According to Figure 3(a), cache hit ratio of the PABL is slightly less than other two schemes. This is because the PABL is a block-based buffer cache schemes. Since the block-based schemes replace all pages of the victim block, it manifests lower cache hit ratio but it can reduce erase overhead as shown in Figure 3(b). According to Figure 3(b), the erase count of the PABL is less than other two schemes. The reason is that the PABL performs block padding like BPLRU to reduce merge overhead when it evicts victim block. Further, if the victim block contains no dirty pages, the PABL evicts the victim block free without write operations to the flash memory. Figure 3(c) shows that the PABL is much better than other two schemes in terms of write count on PRAM. This is because the PABL allocates DRAM buffer for write requests.

5. Conclusion

In this paper, we propose a new buffer management scheme called PABL (PRAM aware Block-based LRU) for mobile computers with DRAM/PRAM hybrid main memory and flash storage devices. Proposed PABL scheme minimizes both the number of write operations on PRAM and the number of erase operations on flash memory. We show through trace-driven

simulation that the PABL outperforms the legacy buffer cache schemes like LRU and CFLRU.

Acknowledgements

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).

References

[1] M. K. Qureshi, V. Srinivasan and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology", Proceedings of International Symposium on Computer Architecture, **(2009)**.

[2] G. Dhiman, R. Ayoub and T. Rosing, "PDRAM: A Hybrid PRAM and DRAM Main Memory System", Proceedings of Design Automation Conference, **(2009)**.

[3] H. Park, S. Yoo and S. Lee, "Power Management of Hybrid DRAM/PRAM-based Main Memory", Proceedings of Design Automation Conference, **(2011)**.

[4] Samsung Electronics, K9XXG08UXM.1G x 8 Bit/2G x 8 bit NAND Flash Memory.

[5] Gal and S. Toledo, "Algorithms and data structures for flash memories", ACM Computing Surveys, 37, 2 **(2005)**.

[6] 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, 48, 2 **(2002)**.

[7] Y. Ryu, "SAT: switchable address translation for flash memory storages", Proceedings of IEEE Computer Software and Applications Conference, **(2010)**.

[8] S. Park, D. Jung, J. Kang, J. Kim and J. Lee, "CFLRU: a replacement algorithm for flash memory", Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, **(2006)**.

[9] Y. Yoo, H. Lee, Y. Ryu and H. Bahn, "Page replacement algorithms for NAND flash memory storages", Proceedings of International Conference on Computational Science and its Applications, **(2007)**.

[10] Z. Li, P. Jin, X. Su, K. Cui and L. Yue, "CCF-LRU: A new buffer replacement algorithm for flash memory", IEEE Transactions on Consumer Electronics, 55, 3 **(2009)**.

[11] H. Jo, J. Kang, S. Park and J. Lee, "FAB: Flash aware buffer management policy for portable media players", IEEE Transactions on Consumer Electronics, 48, 2 **(2006)**.

[12] H. Kim and S. Ahn, "BPLRU: A buffer management Scheme for improving random writes in flash storage", Proceedings of USENIX Conference on File and Storage Technologies, **(2008)**.

Author

Yeonseung Ryu received his BS degree in Computer Science and Statistics from Seoul National University, Korea, in 1990, and his MS and PhD degrees in Computer Science from Seoul National University in 1992 and 1996, respectively. In 1996 he joined Samsung Electronics, Co. as a senior researcher. Since 2003, he has been with Myongji University, Korea, where he is currently a full professor in the Computer Engineering Department. From March 2009 to February 2010, he was a visiting scholar at the University of Minnesota, USA. His research interests include network protocol, network storage systems, and operating systems.