A Memory Management Scheme for Hybrid Memory Architecture in Mission Critical Computers*

Soohyun Yang and Yeonseung Ryu

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

Abstract. Recently as the energy dissipation in mission critical computers are being important issue, non-volatile memory-based hybrid main memory organization have been emerged as a solution. In this paper, we study a new memory management scheme which considers DRAM/PRAM hybrid main memory and flash memory based storages. The goal of proposed scheme is to minimize the number of write operations on PRAM and the number of erase operations on flash memory. In order to evaluate proposed scheme, we compare it with legacy memory management schemes through trace-driven simulation.

Keywords: Buffer Cache, Buffer Replacement, Non-volatile Memory, 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 recent studies introduced PRAM-based main memory organization [1] and DRAM/PRAM hybrid main memory organization [2, 3].

A *buffer cache* mechanism is usually employed in modern operating system (OS) to enhance the performance that is limited by slow secondary storage. When OS receives a read request from an application, file system in OS 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.

* 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).

In this paper, we propose a new buffer cache management scheme called PA-CBF (PRAM-aware Allocation Clean Block First Replacement) for next generation mission critical computers with DRAM/PRAM hybrid main memory and flash memory storage devices. Proposed PA-CBF scheme tries to minimize both the number of write operations on PRAM and the number of erase operations on flash memory. When PA-CBF 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 PA-CBF tries to allocate buffer from PRAM. Otherwise, it tries to allocated buffer from DRAM. During the buffer replacement procedure, PA-CBF considers merge operations performed in FTL in order to minimize the number of erase operations. Trace-driven simulation results show that PA-CBF outperforms the legacy buffer cache schemes.

2 Background and Previous Works

2.1 Flash Memory and Buffer Cache Schemes

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.

If the flash memory is used as storage device, OS usually employs a software module called *flash translation layer* (FTL) between file system and flash memory device [5-6]. An FTL receives read and write requests from the file system and maps a logical address to a physical address in the flash memory. A log block scheme, called block associative sector translation (BAST), was proposed in [5]. 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.

As the flash memory based storage devices have been widely used, buffer cache schemes have also been studied to consider the 'erase-before-write' characteristic of flash memory. The early page-level buffer management schemes did not consider the existence of FTL [7]. They only focused on reducing the number of write operations to flash memory because the write operations accompany the erase operations. Recently some block-level buffer management algorithms consider the address translation algorithm used in FTL to reduce the overhead of garbage collection [8, 9].

In [7], 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. CFLRU can reduce the number of write and erase operations by delaying the flush of dirty pages in the buffer cache.

In [8, 9], 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. 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, which results in decreasing the number erase operations.

2.2 PRAM

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.

3 Proposed Scheme

Fig. 1 illustrates the system configuration considered in this paper in which main memory consists of DRAM and PRAM, and secondary storage is based on flash memory.

The proposed PA-CBF (PRAM-aware Allocation Clean Block First Replacement) scheme is a block-level scheme which maintains a LRU list based on the flash memory block as shown in Fig. 2. 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, the PA-CBF allocates a new buffer and stores page *p* in the allocated buffer. If the block header for block *b* does not exist, the PA-CBF 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.

The PA-CBF defines a *window* as *n* blocks from the LRU position in the list. The size of window is defined as *n*. In Fig 2, for example, window size is 2.

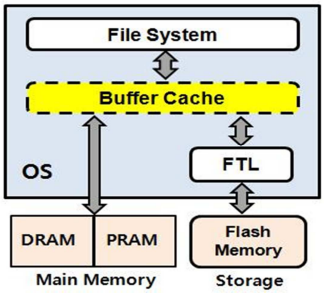


Fig. 1. System configuration.

When the PA-CBF allocates a new buffer, it allocates it from DRAM or PRAM according to the type of request (i.e., read or write). If the request type is read, the PA-CBF tries to allocate buffer from PRAM. Otherwise, it tries to allocate buffer from DRAM. When there is no free space in the desired memory, the PA-CBF allocates it from any type of memory. For example, in Fig. 2, page p12 of block B3 is stored in PRAM buffer because it is requested by read operation.

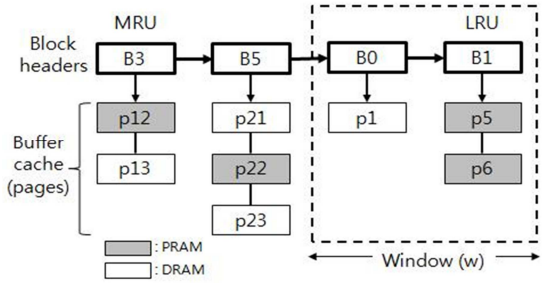
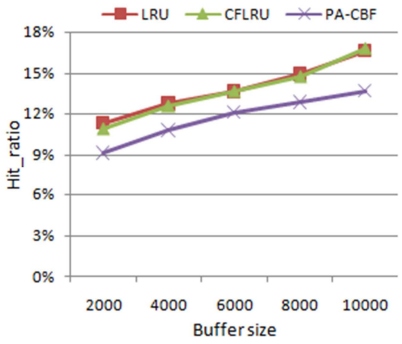


Fig. 2. Buffer cache structure of PA-CBF.

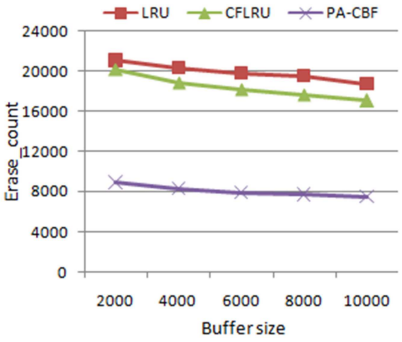
If all free buffers are used up, the PA-CBF must perform buffer replacement procedure. During the buffer replacement procedure, the PA-CBF will find a clean block from the window region (see Fig. 2). A block is clean if member buffers are all clean. If the PA-CBF cannot find a clean block, it selects victim block at LRU position. If the victim block contains dirty pages, then the PA-CBF 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 is clean, the PA-CBF can make the pages of the victim block free without write operations to the flash memory.

4 Simulation Results

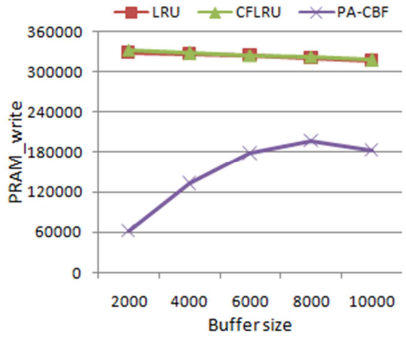
In order to evaluate the proposed scheme, we compared PA-CBF with two of the existing schemes using simulation: LRU and CFLRU. We assume that the hybrid main memory consists of DRAM and PRAM, which are divided by a memory



(a) Hit ratio



(b) Erase count on Flash



(c) Write count on PRAM

Fig. 3. Performance comparison varying buffer cache size

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.

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

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

5 Conclusion

In this paper, we propose a new buffer cache management scheme called PA-CBF for computers with DRAM/PRAM hybrid main memory and flash storage devices. Proposed PA-CBF 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 PA-CBF outperforms the legacy buffer cache schemes like LRU and CFLRU.

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. Samsung Electronics, K9XXG08UXM.1G x 8 Bit/2G x 8 bit NAND Flash Memory
5. 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.
6. Y. Ryu, "SAT: switchable address translation for flash memory storages," in Proc. of IEEE Computer Software and Applications Conference (COMPSAC), Jul. 2010.
7. 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.
8. 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.
9. 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.