A Buffer Management for STT-MRAM based Hybrid Main Memory in Sensor Nodes*

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

Abstract - As the power dissipation has become one of the critical design challenges in a sensor network environment, non-volatile memories such as STT-MRAM and flash memory will be used in the next generation sensor nodes. In this paper, we studied an efficient buffer management scheme considering the write limitation of STT-MRAM based main memory as well as the erase-before-write limitation of flash memory for storage device. The goal of proposed scheme is to minimize the number of write operations on STT-MRAM as well as the number of erase operations on flash memory. We showed through simulation that proposed scheme outperforms legacy buffer management schemes.

Index Terms – Non-volatile memories, Low power, Buffer management, STT-MRAM, NAND flash memory.

1. Introduction

Sensor networks are used in a large number of applications such as military, manufacturing, transportation, safety and environment monitoring. Sensor nodes collect not only sensed data from the environment, but also stream of mass media data like videos and images. In order to perform large data processing, sensor nodes are expected to require much more memory than legacy sensor nodes [1-2]. Some recent studies have shown that DRAM-based main memory spends a significant portion of the total system power [3]. This is a serious problem with battery-powered sensor nodes. Fortunately, the recent advance of memory technology has ushered in new non-volatile memory (NVM) designs such as PRAM (Phase change RAM) and MRAM (Magnetic RAM) that overcome the drawbacks of existing memories such as SRAM or DRAM [4].

Among them, Spin-Torque Transfer Magnetic RAM (STT-MRAM) is being regarded as a promising technology for a number of advantages over the conventional RAMs. STT-MRAM is a next-generation memory that uses magnetic materials as the main information carrier [7]. It achieves lower leakage power and higher density compared to the existing SRAM. Also, STT-MRAM shows higher endurance compared to other NVM techniques such as PRAM or Flash, which makes STT-MRAM more attractive for on-chip memories that must tolerate much more frequent write accesses compared to off-chip memories [7-9].

However, one of the biggest weaknesses of STT-MRAM is long write latency compared to SRAM or DRAM. Since the fast access time of memories on a chip must be guaranteed and cannot be negotiable, the slow write operations of STT-MRAM limit its popularity, even though it shows competitive read performance. Another serious drawback of STT-MRAM is high power consumption in write operations. This issue of high power consumption in STT-MRAM must be resolved due to the limited power budgets.

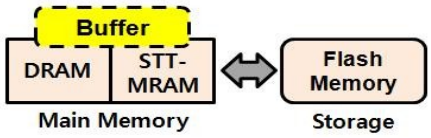


Fig. 1. System configuration.

In order to tackle the energy dissipation in DRAM-based main memory, some studies introduced PRAM-based main memory organization [5], DRAM/PRAM hybrid main memory organization [6], STT-MRAM based main memory organization [10]. Also, there have been some buffer management schemes for PRAM based main memory and DRAM-PRAM hybrid main memory [13]. However, to our knowledge, there is no study on buffer management scheme for STT-MRAM and DRAM hybrid main memory.

In this paper, we study a buffer management scheme for sensor nodes which are equipped with both DRAM/STT-MRAM hybrid main memory and flash memory storages. Fig. 1 illustrates the system configuration considered in this paper. The goal of proposed buffer management scheme is to reduce both the number of write operations on STT-MRAM and the number of erase operations on flash memory. We show that the proposed scheme outperforms other legacy buffer management schemes.

The rest of this paper is organized as follows. In Section 2, we describe the characteristics of STT-MRAM and NAND flash memory. Also, we introduce some buffer management schemes considering non-volatile memory. Section 3, we present a buffer management scheme called Write Aware Buffer (WAB) scheme. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2. Background

A. STT-MRAM

STT-MRAM is a next generation memory technology that takes advantage of magnetoresistance for storing data [7-12].

* 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).

It uses a Magnetic Tunnel Junction (MTJ), the fundamental building block, as a binary storage. An MTJ comprises a three-layered stack: two ferromagnetic layers and an MgO tunnel barrier in the middle (see Fig. 2). Among them, the fixed layer located at the bottom has a static magnetic spin, the spin of the electrons in the free layer at the top is influenced by applying adequate current through the fixed layer to polarize the current, and the current is passed to the free layer. Depending on the current, the spin polarity of the free layer changes either parallel or anti-parallel to that of the fixed layer. The parallel indicates a zero state, and the anti-parallel a one state.

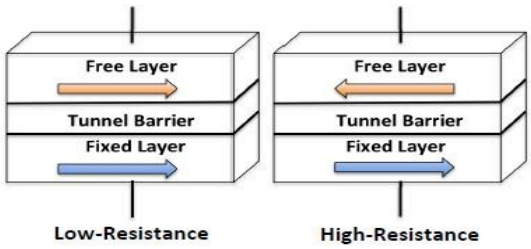


Fig. 2. MTJ block.

Several schemes have been proposed to provide architectural support for applying non-volatile memory to system components. Jog *et al.* [14] proposed to achieve better write performance and energy consumption of STT-MRAM-based L2 cache through adjusting data retention time of STT-MRAM. Similarly, Smullen *et al.* [9] reduced the write latencies as well as dynamic energy of STT-MRAM by lowering the retention time for designing on-chip caches. In [15], they integrated STT-MRAM into on-chip caches in a 3D CMP environment and proposed a mechanism of delaying cache accesses to busy STT-MRAM banks to hide long write latency. Prior to that, Sun *et al.* [16] stacked MRAM-based L2 caches on top of CMPs and reduced overheads through read-preemptive write buffer and hybrid cache design using both SRAM and MRAM. Guo *et al.* [17] resolved the design issues of microprocessors using STT-MRAM in detail for more power-efficient CMP systems.

B. NAND Flash Memory

A NAND flash memory is organized in terms of *blocks*, where each block is of a fixed number of *pages* [18]. A block is the smallest unit of erase operation, while reads and writes are handled by pages. 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. In order to solve erase-before-write problem, a kind of device driver called Flash Translation Layer (FTL) is usually implemented in operating system [19-25]. The FTL performs the physical-to-logical address translation to reduce the number of erase operations. Most address translation schemes use a log block mechanism for storing updates.

A log block scheme, called block associative sector translation (BAST), was proposed by [22]. 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. In BAST, whenever the free log blocks are exhausted, in order to reclaim the log block and the corresponding data block, 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.

C. Buffer Management Schemes

There have been studies on buffer management schemes considering flash memory [26-32]. The flash-aware buffer management schemes can be classified into two categories: page-level [26-30] and block-level schemes [31-32].

A page-level scheme called clean first least recently used (CFLRU) was proposed by [26]. CFLRU maintains a page list by LRU order and divides the page list into two regions, namely the working region and clean-first region like Fig. 3. 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 page cache.

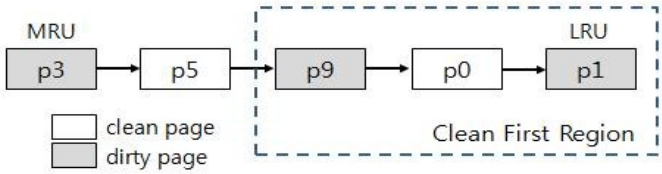


Fig. 3. Page list in CFLRU scheme.

In [32], a block-level buffer cache scheme called block padding LRU (BPLRU) was proposed, which considers the block merge cost in the log block FTL schemes. BPLRU maintains a LRU list based on the flash memory block. 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 BPLRU,

all log blocks can be merged by the switch merge, which results in decreasing the number erase operations.

3. Buffer Management

We propose a buffer management scheme called Write Aware Buffer (WAB). The goal of proposed scheme is to minimize the number of write operations on STT-MRAM as well as the number of erase operations on flash memory.

A. LRU list

The proposed WAB scheme maintains a LRU list like Fig. 4. The LRU list is composed of block headers based on the block of flash memory and each block header manages its own pages loaded from flash memory.

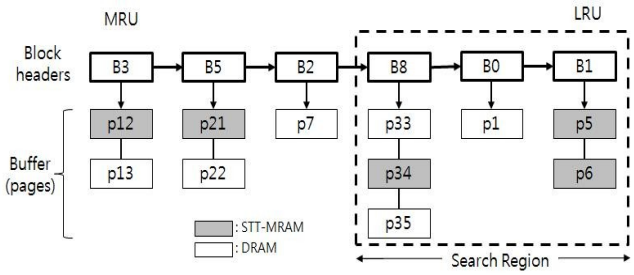


Fig. 4. LRU list based on flash memory block

When a page of a block in the flash memory is first referenced, the WAB allocates a new buffer and stores the requested page in the allocated buffer. If the block header for block does not exist, the WAB allocates a new block header and places it at the MRU position of the LRU list. Then, the WAB attaches the buffer of the requested page to the block header. Whenever a page in the buffer cache is referenced, all pages in the same block are moved to the MRU position.

B. Buffer Allocation and deallocation

We assume that the main memory is divided into DRAM and STT-MRAM by a memory address. The memory which has the low memory address is STT-MRAM and the high section is allocated to DRAM. When WAB allocates a new buffer, it tries to allocate it from the low section (i.e., STT-MRAM).

In order to reduce the number of write operations on STT-MRAM, when a clean page in the STT-MRAM is referenced by a write operation, the WAB allocates a DRAM buffer and writes requested data to the DRAM buffer. Then it deallocates the STT-MRAM buffer. If there is no free DRAM buffer, the WAB frees a clean DRAM buffer from the search region and uses it for storing the requested write data.

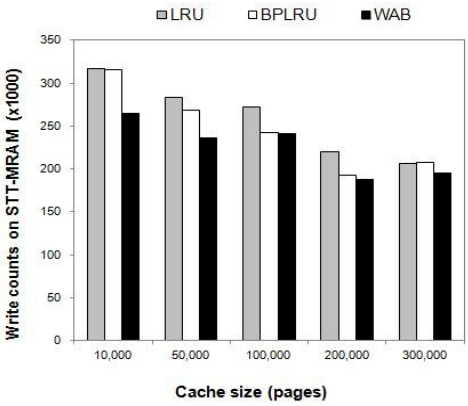
The WAB employs an early deallocation technique which frees clean DRAM buffers early even though free buffers are still available in the system. Because there could be a lot of used buffers that will not be accessed soon, we can free them early with little impact on the cache performance. The WAB searches clean blocks which use only DRAM buffers from the search region periodically or whenever the number of free DRAM buffers falls down below a threshold. Then, it frees them. This technique can decrease the number of writes on STT-MRAM because the WAB can secure free DRAM buffers for new allocations.

If all buffers are used up, the WAB selects a victim block from the search region. In order to reduce the number of erase operations on flash memory, the WAB tries to find a clean block and simply frees all pages in it. If there is no clean block in the search region, the WAB selects a victim block at the LRU position of the block list, performs the page padding technique [32], and flushes all pages of the victim block.

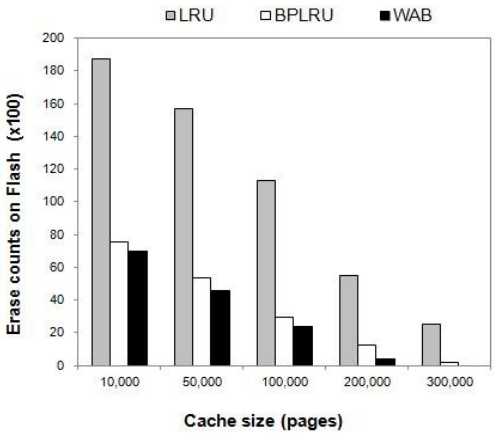
4. Experiment Results

In order to evaluate the proposed scheme, we have developed a trace-driven simulator. For the workload, we obtained buffer I/O traces from a laptop for a week. The total I/O count is 706,833 and I/O ratio is about 55:45.

Fig. 5 (a) shows that the WAB outperforms other schemes in terms of the write counts on STT-MRAM. The WAB reduces write counts by roughly 13% on average and up to 17%. In Fig. 5 (b), the WAB can dramatically reduce the erase counts on flash memory as BPLRU does. Further, the WAB outperforms BPLRU because it considers clean blocks to avoid erase operation during replacement procedure.



(a) Write counts on STT-MRAM.



(b) Erase counts on Flash.

Fig. 5. Performance evaluation result.

5. Conclusion

Recently, in order to tackle the energy dissipation in DRAM-based main memory, there have been some studies which consider next generation non-volatile memories such as PRAM and STT-MRAM as main memory instead of DRAM.

In this paper, we study a buffer management scheme called WAB for sensor nodes which have DRAM/STT-MRAM hybrid main memory and NAND flash memory storage. The proposed buffer management scheme minimizes both the number of write operations on STT-MRAM and the number of erase operations on flash memory. We showed through trace-driven simulation that proposed scheme outperforms legacy buffer cache schemes. For the future work, we will evaluate our scheme by using real traces of sensor nodes.

6. References

[1] N. Lin, Y. Dong, and D. Lu, "Providing virtual memory support for sensor networks with mass data processing," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[2] A. Lachemann, P. Marron, M. Gauger, D. Minder, O. Saukh, and K. Rothermel, "Removing the memory limitations of sensor networks with flash-based virtual memory," *SIGOPS Operating Systems Review*, vol. 41, pp. 131-144, 2007.

[3] L. Barroso, and U. Holzle, "The case for energy-proportional computing," *IEEE Computer*, vol.40, no.12, 2007.

[4] X. Yuan, "Modeling, architecture, and applications for emerging memory technologies," *IEEE Design & Test of Computers*, vol. 28, no. 1, pp. 44-51, 2011.

[5] M. Qureshi, V. Srinivasan, and J. Rivers, "Scalable high performance main memory system using phase-change memory technology," In *Proc. of International Symposium on Computer Architecture*, 2009.

[6] H. Park, S. Yoo, and S. Lee, "Power management of hybrid DRAM/PRAM-based main memory," In *Proc. of Design Automation Conference*, 2011.

[7] H. Jang, B. An, N. Kulkarni, K. Yum, E. Kim, "A hybrid buffer design with STT-MRAM for on-chip interconnects," in *Proc. of ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2012.

[8] S. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy, "Future cache design using STT MRAMs for improved energy efficiency: devices, circuits and architecture," in *Proc. of Design Automation Conference (DAC)*, pp. 492-497, 2012.

[9] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *Proc. of High Performance Computer Architecture (HPCA)*, pp. 50-61, 2011.

[10] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013.

[11] A. Charles, N. Mojumder, X. Fong, S.Choday, S. Park, and K. Roy, "Spin-Transfer torque MRAMs for low power memories: perspective and prospective," *IEEE Sensors*, vol. 12, no. 4, pp. 756-766, 2012.

[12] D. Lee, S. Gupta, and K Roy, "High-performance low-energy STT MRAM based on balanced write scheme," in *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 9-14, 2012.

[13] H. Seok, Y. Park, and Park, K., "Efficient page caching algorithm with prediction and migration for a hybrid main memory," *Applied Computing Review*, vol. 11, no. 4., 2012.

[14] A. Jog, A. K. Mishra, C. Xu, Y. Xie, N. Vijaykrishnan, R. Iyer,and C. R. Das, "Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPs," *The Pennsylvania State University CSE Dept., Tech. Rep. CSE-11-010*, June 2011.

[15] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das, "Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs," in *Proc. of ISCA*, 2011.

[16] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc. of HPCA*, 2009.

[17] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing," in *Proc. of ISCA*, 2010.

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

[19] Intel Corp. "Understanding the flash translation layer (FTL) specification," 1998.

[20] Y. Ryu, "Method for controlling flash memory device," Korea Patent 10-0638638, Sep. 2004.

[21] Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, vol. 37, no. 2, 2005.

[22] 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.

[23] Gupta, Y. Kim, and B. Urgaonkar, "DFTL: a flash translation layer employing demand-based selective caching of page-level address mapping," in *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems*, 2009, pp.229-240.

[24] Y. Ryu, "SAT: switchable address translation for flash memory storages," in *Proc. of IEEE Computer Software and Applications Conference (COMPSAC)*, Jul. 2010.

[25] M. Chiang and R. Chang, "Cleaning policies in mobile computers using flash memory," *Journal of Systems and Software*, vol. 48, no. 3, pp. 213–231, 1999.

[26] 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.

[27] Y. Yoo, H. Lee, Y. Ryu, and H. Bahn, "Page replacement algorithms for NAND flash memory storages," in *Proc. of International Conference on Computational Science and its Applications*, pp. 201-212, 2007.

[28] 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*, vol. 55, no. 3, pp. 1351-1359, August, 2009.

[29] J. Park, H. Lee, S. Hyun, K. Koh, and H. Bahn, "A cost-aware page replacement algorithm for NAND flash based mobile embedded systems," in *Proc. of EMSOFT*, pp. 315-324, 2009.

[30] X. Tang and X. Meng, "ACR: An adaptive cost-aware buffer replacement algorithm for flash storage devices," in *Proc. of 11th International Conference on Mobile Data Management*, pp. 33-42, 2010.

[31] 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.

[32] 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.