Block-level Replacement Scheme Considering Re-write Probability for Solid State Drives

Ilhoon Shin

Electronic and IT Media Engineering Department, Seoul National University of Science and Technology
ilhoon.shin@snut.ac.kr

Abstract

SSDs use multiple NAND flash memory chips as storage media and deploy large sized RAM inside it in order to maintain the FTL mapping table. The rest portion of the inner RAM can be used as buffer. The buffer absorbs the read/write requests by file systems and thus the resulting write requests to NAND flash memory is determined by the buffer replacement scheme. The block-level LRU replacement schemes, which manages the buffer in NAND block unit, generates a large sized write pattern that is NAND-friendly. However, the existing schemes do not consider the re-write probability of each page. This paper presents a new block-level replacement policy for SSDs. The presented scheme evicts only cold pages that its re-write probability is low considering the re-write probability, which can contribute to improve the buffer hit ratio.

Keywords: *re-write probability, buffer replacement, Solid State Drives, NAND flash memory*

1. Introduction

Solid State Drives (SSDs), which consists of multiple NAND flash memory chips, has been emergently used in laptop, PC, and server markets due to the strengths of fast read performance, light-weight, low energy consumption, small form factor, and shock-resistance. However, the write performance is relatively slow compared to the read performance, and thus many previous researches have been tried to improve the write performance of SSDs. The use of internal RAM (or Non-volatile RAM) inside SSDs is one of those approaches [1-3, 7-8].

SSDs use NAND flash memory as storage media. NAND flash memory is a kind of EEPROM (Electrically Erasable Programmable Read Only Memory). It consists of blocks and pages. A block is generally 256 KB in size, and a page is generally 2 KB or 4 KB. In NAND flash memory, data are written in the unit of a page similarly to hard disk drives. However, NAND flash memory does not support an overwrite operation. In order to write new data, the page should be first erased. The erase operation is order of magnitude slower than the read and the write operations, and the unit of the erase operation is a block, which is larger than the page. Erasing the block clears the data of the other pages of the block. Thus, the overwrite operation is performed in the way of the out-of place update by flash translation layer (FTL). When writing the data, FTL searches for a clean page and write the data to the clean page. At this time, the old page is invalidated. In the out-of place update, the location of the data becomes different on every overwrite operation. Thus, FTL maintains the mapping table between the logical sector number and its physical location. Meanwhile, the clean pages will be scarce by the continuous write requests, and it triggers a garbage collection process.

The garbage collection process selects the victim block for erasing. The valid pages of the victim block are copied to the other clean pages, and the victim block is erased. The garbage collection process causes multiple page write operations and a block erase operation at least, which hurts the performance of NAND-based storage seriously. Thus, the previous researches have tried to design an efficient sector mapping schemes for FTL to reduce the frequency and the latency of the garbage collection process [4-5, 9-10].

In the meantime, using the internal RAM of SSDs can contribute to improve the performance of SSDs by absorbing the write requests with the buffer. The write requests can be served by the buffer without writing to the NAND flash memory. If the buffer is full, the victim buffer is replaced. The victim buffer is written to the NAND flash memory if the victim buffer is dirty. Thus, if the buffer hit ratio is high and the write pattern caused by the buffer replacement is NAND-friendly, the performance of SSDs will be improved. The goal of the paper is to design NAND-friendly buffer replacement scheme for SSDs. The new policy tries to increase the buffer hit ratio while at the same time generating the NAND-friendly write pattern.

The rest of the paper is organized as follows. Section 2 explains the related work focusing on the strengths and the drawbacks of the previous buffer replacement schemes. Section 3 describes the new buffer replacement scheme and section 4 evaluates its performance. Finally, section 5 draws a conclusion.

2. Related Work

The existing buffer replacement schemes for SSDs are classified to page-level and block-level schemes. The page-level schemes manage the buffer in a NAND page unit. In order to increase the hit ratio, they adopt the LRU (Least Recently Used) replacement scheme. On a buffer hit, the hit page is moved to the head of the LRU list. On a scarce of the buffer, the last page is replaced. If the page is dirty, it is written to NAND flash memory. Thus, the page-level schemes cause the small sized random write pattern, which hurts the performance of the NAND-based storages [6]. The CFLRU (Clean First LRU) scheme [7] is a kind of the page-level LRU replacement scheme. The difference is that it replaces the unmodified pages first to reduce the write operations to the NAND flash memory.

The block-level replacement schemes manage the buffer in a NAND block unit. The FAB (Flash Aware Buffer) scheme [1], which is a kind of the block-level replacement scheme, replaces the block buffer that has the most pages. Thus, it causes a large sized write pattern, which is NAND-friendly. However, the buffer hit ratio is low because it does not consider the temporal locality.

The block-level LRU replacement scheme (BLRU) [2, 8] maintains the block buffers with the LRU list. If a page is hit, the block buffer that the page belongs to is moved to the head of the LRU list. On the scarce of the buffer space, the last block buffer of the LRU list is replaced. Thus, it generates a large sized write pattern while at the same time improving the buffer hit ratio by considering the temporal locality. The weakness is that the rarely access page that belongs to the frequently accessed block occupy the buffer for a long time.

The BPLRU (Block Padding LRU) scheme [8] is a kind of the block-level replacement scheme. Similarly to the BLRU scheme, it maintains the block buffer with the LRU list. The different is that it pads empty pages of the victim block buffer on the eviction and always writes the whole block. Thus, the BPLRU scheme guarantees the block sized write pattern. However, the padding accompanies a considerable overhead when the victim block buffer is almost empty.

The PLRU-BR (Page-level LRU & Block Replacement) scheme [3] mixed the block-level LRU and the page-level LRU schemes. It maintains the LRU list in a page unit like the page-level LRU schemes. If a page is hit, it is moved to the head of the LRU list. The difference from the page-level LRU schemes is that it replaces all the pages that belong to the same block with the victim page. Thus, the PLRU-BR scheme causes a sequential write pattern like the block-level LRU scheme. However, the frequently accessed pages that belong to the same block with the victim page can be replaced from buffer, which hurts the buffer hit ratio.

3. Block-level Replacement Scheme Considering a Re-write Probability

In order to address the weaknesses of the previous block-level replacement schemes, we design a new block-level replacement scheme, PLRU-BR-2Q (Queue). The PLRU-BR-2Q scheme is similar to the PLRU-BR scheme. The difference is that on the buffer replacement it evicts only the cold pages which are not likely to be re-written later. The hot pages that belong to the same block with the victim page remain in buffer.

In order to evaluate the re-write probability of each page, the PLRU-BR-2Q scheme maintains two LRU lists: hot list and cold list. The size of the hot list is fixed to n % of the entire buffer. On a write request, the requested page is moved to the head of the hot list. Because the size of the hot list is fixed, the last page of the hot list is moved to the head of the cold list if the hot list is full. When the buffer is full, the last page of the cold list becomes a victim page, and the cold pages which reside in cold list and belong to the same block with the victim page are evicted from the buffer. The hot pages which reside in hot list are not replaced even if they belong to the same block with the victim page. Thus, the PLRU-BR-2Q scheme can improve the buffer hit ratio by keeping the hot pages in the buffer.

4. Performance Evaluation

In order to evaluate the performance of the presented buffer replacement scheme, we used a trace-driven simulation. In the PLRU-BR-2Q scheme, the size of the hot list is 10 % of the entire buffer. The latencies of read, write, and erase operations of a NAND page and a NAND block are assumed as 25 us, 200 us, 2 ms, respectively. The target SSD is assumed as 2-channel & 4-way structure, and a clustered page is assumed to be 16KB in size under the assumption that a physical NAND page is 2KB in size. Similarly, a clustered block is assumed to be 1 MB under the assumption that a physical block is 128 KB in size. Transferring 2KB data via each channel is assumed to take 70 us. The main performance measure is the total elapsed time, which is calculated using the following formula: (total elapsed time = page read count $\times$ page read latency + page write count $\times$ page write latency + block erase count $\times$ block erase latency + buffer access count * buffer access latency). We used two realist workloads which were collected in a PC formatted with NTFS file system. The partition size of the NTFS1 trace is 32 GB, and the partition size of the NTFS2 trace is 68 GB.

Figure 1 shows the total elapsed time of each buffer replacement schemes when the BAST scheme is used as the FTL sector mapping scheme. The X axis is the buffer size, which varies from 0 MB to 256 MB. The result of 0 MB of buffer space is the performance of the BAST scheme itself without the buffer. Y axis is the total elapsed time in seconds. We excluded the result of the BPLRU scheme because it delivered a 4 times worse performance than the others due to the excessive padding overhead. From the figure, we can see that the FAB scheme is worse than the other LRU-based replacement schemes. This is because it does not consider the temporal locality and thereby its buffer hit ratio is seriously low. Among the LRU-based replacement schemes, the block-level replacement schemes were better than the page-level replacement scheme (CFLRU). The block-level replacement schemes generate a larger sized write pattern than the CFLRU scheme, which is NAND-friendly and thus delivers a better performance. The performance improvement by discerning hot pages from cold pages was not conspicuous in the used NTFS traces.

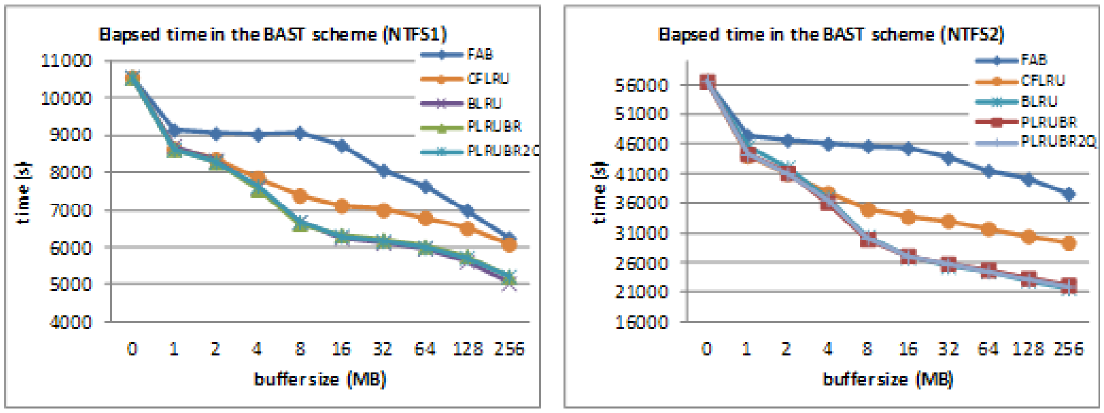

Figure 1. Elapsed Time varying buffer size (BAST scheme in NTFS1 & NTFS2)

Figure 2 shows the total elapsed time of each buffer replacement schemes when the FAST scheme is used as the FTL sector mapping scheme. The result of the BPLRU scheme was excluded due to its seriously worse performance. The result of the FAST scheme is similar to the result of the BAST scheme. The block-level LRU replacement schemes delivered a better performance than the other schemes. Meanwhile, the PLRU-BR-2Q scheme was worse than the other block-level LRU replacement schemes in the NTFS2 trace.

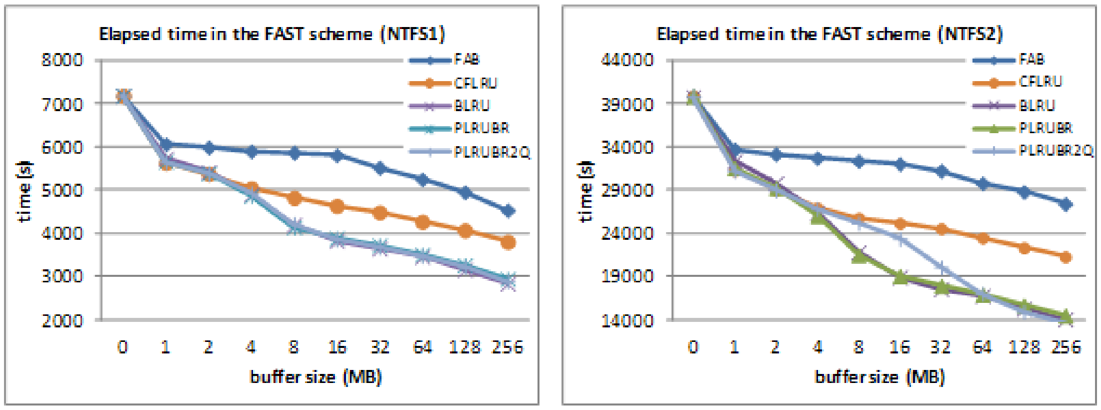

Figure 2. Elapsed Time varying buffer size (FAST scheme in NTFS1 & NTFS2)

Figure 3 shows the total elapsed time of each buffer replacement schemes when the page mapping (PMAP) scheme is used as the FTL sector mapping scheme. In the page mapping scheme, the page-level LRU replacement scheme (CFLRU) delivered a slightly worse performance than the block-level LRU replacement schemes. The performance of the PLRU-BR-2Q scheme was similar to the other block-level replacement schemes.

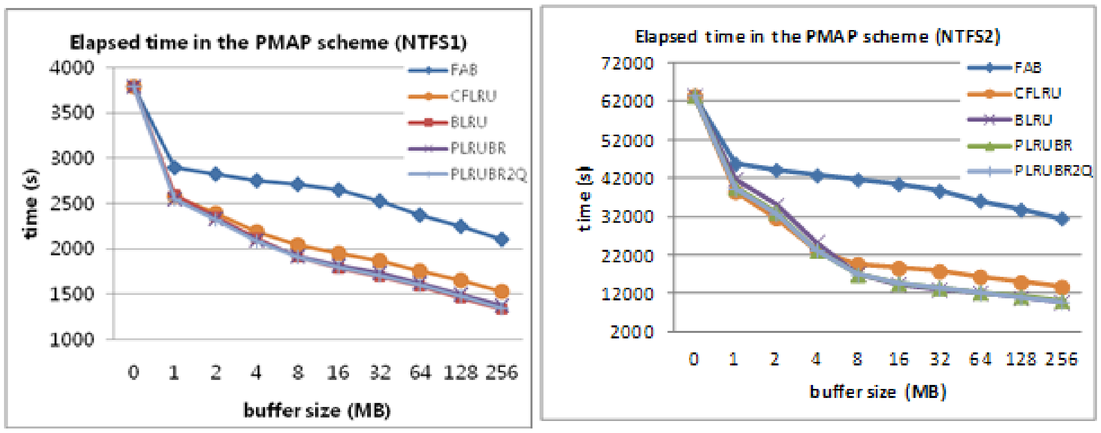

Figure 3. Elapsed time varying buffer size (PMAP scheme in NTFS1 & NTFS2)

5. Conclusion

In this paper, we presented a new block-level buffer replacement policy, PLRU-BR-2Q for SSDs. The PLRU-BR-2Q scheme performed the NAND block-level replacement considering the temporal locality in order to generate a large sized write pattern. Also, in order to improve the buffer hit ratio, it discerned hot pages from cold pages. The hot pages were not evicted even though they belonged to the same block with the victim page. The simulation result using the PC traces showed that the performance improvement from discerning hot pages from cold pages was not conspicuous compared to the block-level replacement. However, in the trace where hot pages and cold pages are clearly discerned such as database, the PLRU-BR-2Q scheme would deliver a better performance than the other block-level LRU replacement schemes. Our future work is to perform a similar evaluation in other kinds of traces in order to investigate the effect of discerning hot pages from cold pages.

Acknowledgements

This work was supported by Seoul National University of Science and Technology and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0003938).

References

[1] H. Jo, J. Kang, S. Park, J. Kim and J. Lee, “FAB: Flash-aware buffer management policy for portable media players”, IEEE Transactions on Consumer Electronics. Vol. 52, pp. 485-493 **(2006)**.

[2] S. Kang, S. Park, H. Jung, H. Shim and J. Cha, “Performance Trade-offs in using NVRAM write buffer for flash memory-based storage devices”, IEEE Transactions on Computers. Vol. 58, No. 6, pp. 744-758 **(2009)**.

[3] I. Shin, “Performance evaluation of buffer replacement schemes for solid state drives”, Lecture Notes in Electrical Engineering. Vol. 142, pp. 481-488 **(2012)**.

[4] J. Kim, J. M. Kim, S. Noh, S. Min and Y. Cho, “A space-efficient flash translation layer for compactflash systems”, IEEE Transactions on Consumer Electronics. Vol. 48, pp. 366-375 **(2002)**.

[5] S. Lee, D. Park, T. Chung, W. Choi, D. Lee, S. Park and H. Song, “A log buffer based flash translation layer using fully associative sector translation”, ACM Transactions on Embedded Computing Systems. Vol. 6, No. 3 **(2007)**.

[6] I. Shin, “Influence of access pattern on performance of NAND-based storage”, Communications in Computer and Information Science. Vol. 194, pp. 63-69 **(2011)**.

[7] S. Park, D. Jung, J. Kang, J. Kim and J. Lee, “CFLRU: A replacement algorithm for flash memory”, Proceedings of International Conference of Compilers, Architecture, and Synthesis for Embedded Systems, **(2006)**.

[8] H. Kim and S. Ahn, “BPLRU: A buffer management scheme for improving random writes in flash storage”, Proceedings of USENIX FAST, **(2008)**.

[9] A. Ban, “Flash file system”, U.S. Patent 5,404,485 **(1995)**.

[10] A. Ban, “Flash file system optimized for page-mode flash technologies”, U.S. Patent 5,937,425 **(1999)**.

Authors

Ilhoon Shin

Ilhoon Shin received the B.S., the M.S., and the ph.D degrees in computer science and engineering from Seoul National University, Korea. He is currently an assistant professor of the department of electronics and information engineering at Seoul National University of Science & Technology. His research interests include storage systems, embedded systems, and operating systems.