HLRU: Hybrid Buffer Replacement Scheme for Solid State Drives

Ilhoon Shin

Seoul National University of Science & Technology

ilhoon.shin@snut.ac.kr

Abstract

This paper presents buffer replacement schemes for Solid State Drives (SSDs). SSDs consist of multiple NAND flash memory chips, and deploy internal RAM or NVRAM to maintain the mapping table of flash translation layer. The available internal RAM can be used as buffer to absorb a portion of the read/write requests. In order to maximize the effect of the buffer, we have to increase the buffer hit ratio with generating NAND-friendly access pattern. The presented buffer replacement schemes prioritize hot pages that are likely to be re-written in order to increase the buffer hit ratio. In order to generate NAND-friendly write pattern, the presented scheme perform a replacement in a NAND block unit, because NAND-based storage delivers a good performance when the average write request is large in size. The trace-drive simulation shows that the presented schemes deliver a good performance.

Keywords: *Hot pages, LRU, Buffer replacement, Solid State Drives, NAND flash memory*

1. Introduction

Nowadays, Solid State Drives (SSDs) have been drastically used in notebooks, desktop computers, and servers. SSDs, which consist of multiple NAND flash memory chips, have 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. Thus, many studies have been performed to improve the write performance of SSDs. One approach is to use internal RAM (or Non-volatile RAM) inside SSDs as buffer [1-4, 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 128 KB or 256 KB in size, and a page is 2 KB or 4 KB in size. In NAND flash memory, data are written in the unit of a page similarly to hard disk drives. The difference is that NAND flash memory does not support an overwrite operation. In order to over-write new data, the target page should be erased. The problem is that the erase is performed in the unit of a block. If we erase the block that the target page belongs to, the data of the other pages are also erased. Thus, the over-write operation cannot be supported in a way of in-place update. Instead, NAND flash memory deploys the out-of place update, which writes new data to another clean page. Supporting the over-write operation with the out-of-place update is the main function of flash translation layer (FTL).

On write requests, FTL searches for a clean page and write the data to the found page. The original page is invalidated at the time. Thus, the continuous out-of-place update will eventually exhaust clean pages, which initiate a garbage collection process. The garbage collection process selects the victim block and moves the valid pages of the victim block to other clean block. Finally, the victim block is erased and the clean pages are reclaimed. Because the garbage collection process generally accompanies multiple page copies and

block erasures, it is crucial for the performance to reduce the frequency of the garbage collection process. Thus, lots of previous researches have focused on designing an efficient FTL schemes to reduce the frequency of the garbage collection process [5-8].

Meanwhile, in NAND-based storages such as SSDs where the large sized RAM or NVRAM is deployed, a portion of the read/write requests can be served without accessing NAND flash memory by using the internal RAM as the buffer, which can contribute to improve the performance of SSDs. In order to maximize the effect of the buffer, we have to increase the buffer hit ratio with generating NAND-friendly access pattern. In order to increase the buffer hit ratio, the buffer replacement scheme should consider the re-reference probability. Because page read latency is much shorter than page write latency in NAND flash memory, we mainly focus on the re-write probability. Also, NAND-based storage delivers a good performance when the average write request is large in size. Thus, we deploy a block-level replacement, which performs a replacement in a NAND block unit.

The rest of the paper is organized as follows. Section 2 explains the related work. Section 3 describes the presented buffer replacement scheme, and section 4 presents its performance evaluation result. 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 of the LRU list is replaced. If the page is dirty, it is written to NAND flash memory. Thus, the page-level schemes generate the small sized random write pattern, which hurts the performance of the NAND-based storages. The CFLRU (Clean First LRU) scheme [7] is a kind of the page-level LRU replacement scheme. It replaces the unmodified pages first to reduce the number of the write operations to 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 the temporal locality is not considered.

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 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 accessed pages that belong to the frequently accessed block remain in the buffer for a long time.

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

The PLRU-BR (Page-level LRU & Block Replacement) scheme [3] mixes the BLRU scheme and the page-level LRU scheme. It maintains the LRU list in a page unit like the page-level LRU scheme. If a page is hit, it is moved to the head of the LRU list. The difference is that it replaces all the pages that belong to the same block with the victim

page on the page eviction. Thus, the PLRU-BR scheme generates a large sized write pattern like the BLRU scheme. The weakness is that the frequently accessed pages that belong to the same block with the victim page can be replaced from the buffer, which hurts the buffer hit ratio.

The PLRU-BR-2Q (Queue) scheme [4] is similar with the PLRU-BR scheme. It maintains the page-level LRU list and selects the victim page by the LRU replacement. The pages that belong to the same block with the victim page are replaced together. The difference is that the hot pages, which are likely to be re-written, remain in the buffer even though they belong to the same block. Only cold pages, which are not likely to be re-written, are replaced from the buffer. The weakness of the PLRU-BR-2Q scheme is that the average request size is smaller than the BLRU scheme and the PLRU-BR scheme, because it evicts only the cold pages. The small sized write requests hurt the sequentiality of the write pattern and may damage the performance of the underlying FTL scheme.

3. HLRU Scheme

In order to compromise the drawbacks of the existing schemes, the hybrid block-level LRU (HLRU) scheme operates the BLRU scheme and the PLRU-BR-2Q scheme together. It maintains three LRU lists, the block-level LRU list, the page-level hot list, and the page-level cold list. The block-level LRU list is managed with the same way with the BLRU scheme. When the page is accessed, the block that the page belongs to is moved the head of the block-level LRU list. The page-level hot list and the cold list are managed in the same way with the PLRU-BR-2Q scheme. The hot list maintains the recently written pages, and its size is fixed to n % of the total buffers. The cold list maintains the unwritten pages for a long time.

The HLRU scheme basically operates the BLRU replacement scheme. When the buffer is full, it selects the last block of the block-level LRU list as victim and evicts the victim block. In order to address the weakness of the BLRU scheme that the cold pages that belong to the hot block may remain in the buffer for a long time, the HLRU scheme sometimes selects the last page of the page-level cold list as victim and evicts all the cold pages that belong to the same block with the victim page, which is the same with the PLRU-BR-2Q scheme. The ratio of performing the PLRU-BR-2Q scheme is fixed to 10 %. The HLRU scheme can make the average write request size the same with the BLRU scheme while at the same time mitigating the weakness of the BLRU scheme with the help of the PLRU-BR-2Q scheme.

4. Performance Evaluation

We evaluate the performance of the presented buffer replacement scheme through a trace-driven simulation. Traces were collected in PCs that formatted with NTFS file system. The partition size of the NTFS1 trace is 32 GB, and the total amount of read bytes and the total amount of written bytes are 16 GB and 35 GB, respectively. The partition size of the NTFS2 trace is 68 GB, and the total amount of read bytes and the total amount of written bytes are 201 GB and 187 GB, respectively.

The target SSD is assumed to connect multiple NAND flash memory chips with 2-channel & 4-way structure. A physical NAND page is 2 KB in size, and reading and writing a page takes 25 us and 200 us, respectively. A physical block is 128 KB in size, and erasing a block takes 2 ms. Transferring 2KB data via each channel is assumed to take 70 us.

In 2-channel & 4-way structure, 8 physical pages are read/written together, and they are regarded as one clustered page [9]. Similarly, 8 physical blocks are erased together, and they

are regarded as one clustered block. Thus, a clustered page is 16 KB in size, and a clustered block is 1 MB in size. The performance measure is the total I/O time, which is calculated using the following formula: (total I/O time = clustered page read count × clustered page read latency + clustered page write count × clustered page write latency + clustered block erase count × clustered block erase latency + buffer access count * buffer access latency). The clustered page read/write latency is calculated from the physical page read/write latency and the channel transfer time [9]. The clustered block erase latency is almost the same with the physical block latency because the latency of transferring the erase command is negligible.

The performance of the presented buffer replacement schemes are compared with the representative buffer replacement schemes for SSDs. The size of the hot list is 10 % of the entire buffer, and the ratio of performing the BLRU scheme and the PLRU-BR-2Q scheme is 9 to 1.

Figures 1 and 2 show the performance evaluation result in NTFS1 and in NTFS2, when the BAST scheme [5] is used as the FTL sector mapping scheme. The X axis of the figures is the buffer size, which varies from 0 MB to 128 MB. 0 MB means that the buffer is not used. Y axis is the total I/O time of performing the trace file in seconds. Among the representative buffer replacement schemes, the result of the BPLRU scheme is excluded, because it delivered a much worse performance than the other schemes due to the excessive padding overhead. The result shows that the FAB scheme delivers a much worse performance than the other LRU-based schemes in the both traces, because the temporal locality is not considered. The CFLRU scheme is worse than the other block-level LRU replacement scheme especially when the buffer is large. It is because the CFLRU scheme, which is a page-level LRU replacement scheme, causes a small sized random write pattern that is not NAND-friendly. The block-level LRU replacement schemes (BLRU, PLRU-BR-2Q, and HLRU) deliver the similar performance in the both traces. The performance improvement of prioritizing hot pages is not conspicuous in the used NTFS traces.

Figures 3 and 4 show the performance evaluation result when the FAST scheme [6] is used as the FTL sector mapping scheme. The result of the BPLRU scheme is also excluded because of 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 deliver a better performance than the other schemes. Although, the FAST scheme copes well with the small sized random write pattern than the BAST scheme, the CFLRU scheme is still worse than the block-level LRU replacement schemes. Among the block-level replacement schemes, the PLRU-BR-2Q scheme delivers a worse performance than the BLRU scheme and the HLRU scheme in the NTFS2 trace.

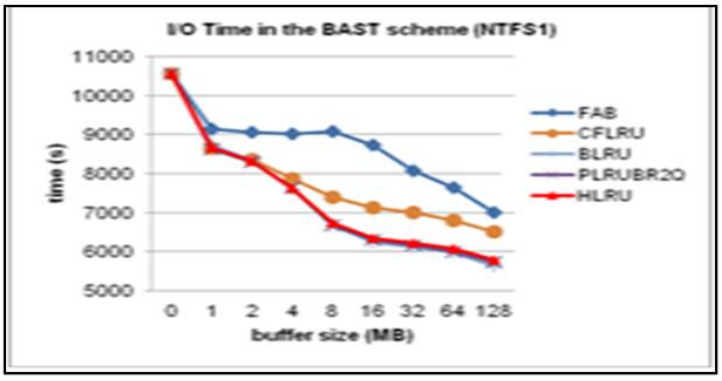

Figure 1. Total I/O Time in the BAST (NTFS1)

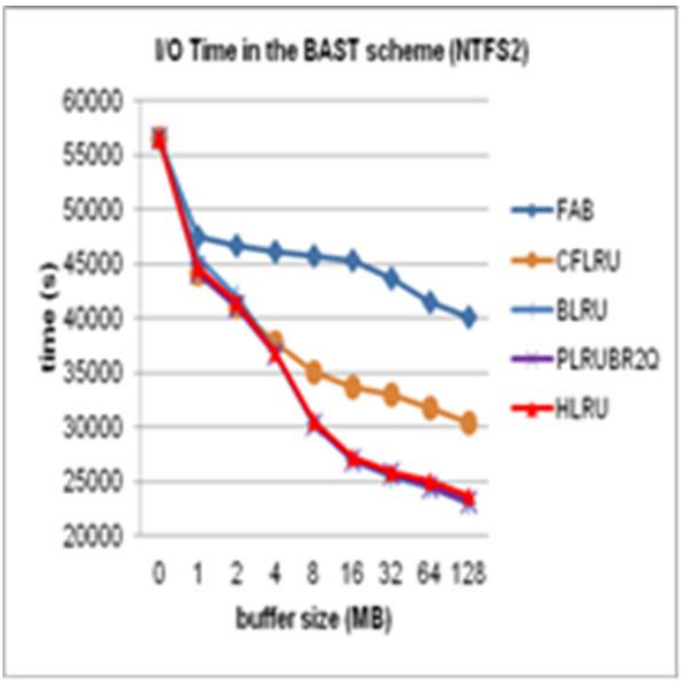

Figure 2. Total I/O Time in the BAST (NTFS2)

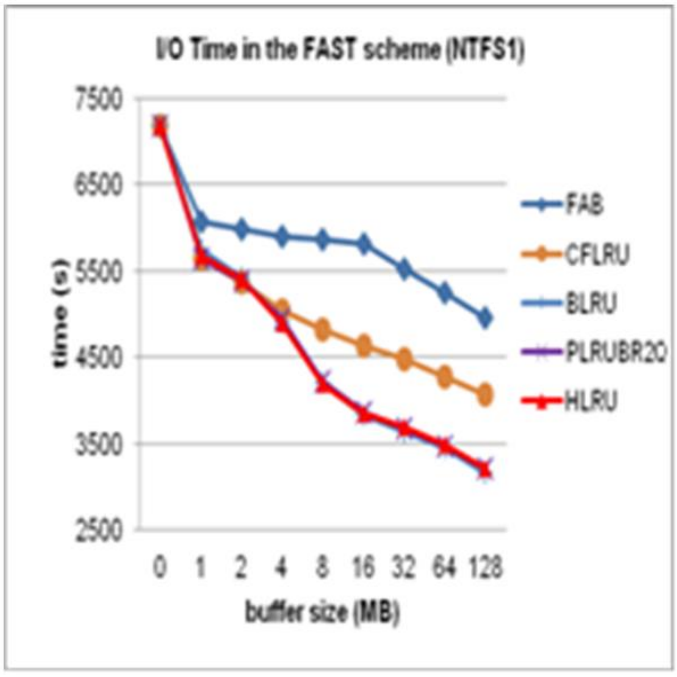

Figure 3. Total I/O Time in the FAST (NTFS1)

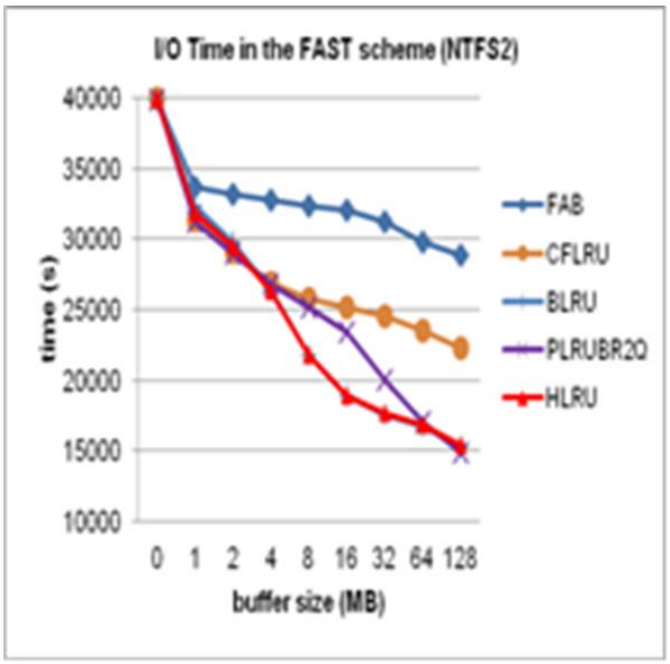

Figure 4. Total I/O Time in the FAST (NTFS2)

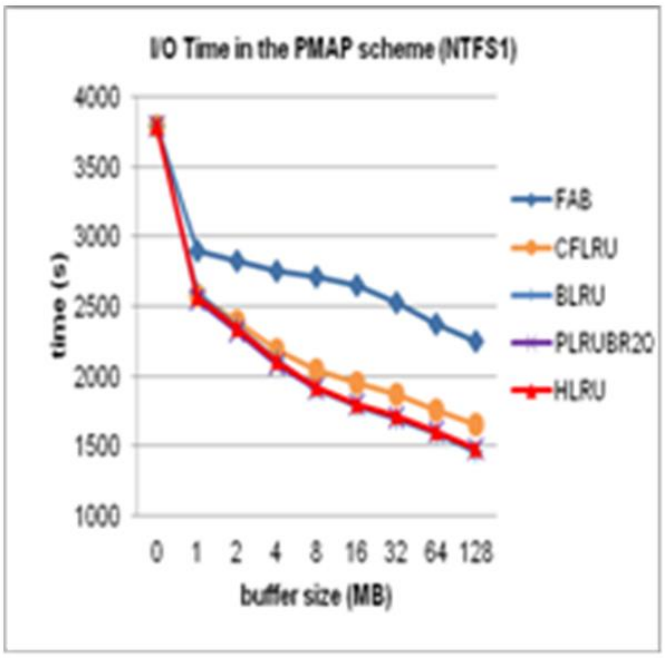

Figure 5. Total I/O Time in the PMAP (NTFS1)

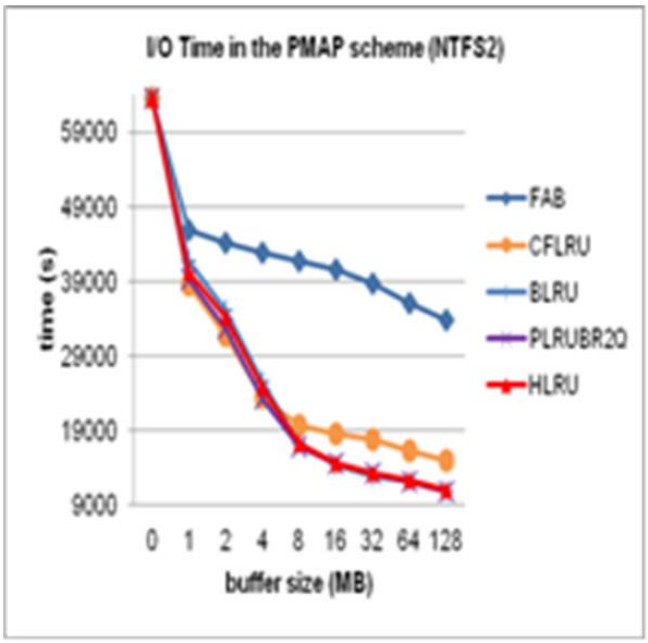

Figure 6. Total I/O Time in the PMAP (NTFS2)

Figures 5 and 6 show the performance evaluation result when the page mapping scheme [11] is used as the FTL sector mapping scheme. In the page mapping scheme, the page-level LRU replacement scheme (CFLRU) delivers a slightly worse performance than the other block-level LRU replacement schemes, because the page mapping scheme handles the small sized random write pattern the best. The performance improvement of prioritizing hot pages is not conspicuous.

5. Conclusion

In this study, we presented new block-level buffer replacement policies for SSDs. The HLRU scheme operated both the BLRU scheme and the PLRU-BR-2Q scheme. Basically, it operated the BLRU scheme in order to increase the average write request size. However, it selected a victim with the PLRU-BR-2Q scheme once in ten times in order to evict the cold pages that belonged to the hot block.

The simulation result using the PC traces showed that the performance improvement was not significant compared to the gain of the large write requests. The HLRU scheme delivered a similar performance with the existing block-level replacement schemes in the PC traces. However, in the workloads such as database where the hot pages and the cold pages are more clearly identified, the presented schemes would deliver a better performance. Our future work is to evaluate the effect of prioritizing the hot pages in other workloads.

Acknowledgements

This study was financially 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, **(2006)**, pp. 485-493.
[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, **(2009)**.
[3] I. Shin, "Performance evaluation of buffer replacement schemes for solid state drives", Lecture Notes in Electrical Engineering, vol. 142, **(2012)**.
[4] I. Shin, "Block-level Replacement Scheme Considering Re-write Probability for Solid State Drives", SERSC International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, **(2012)**.
[5] 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, no. 2, **(2002)**.
[6] 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)**.
[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] J. Kim, D. Jung, J. Kim and J. Huh, "A methodology for extracting performance parameters in Solid State Disks (SSDs)", Proceedings of MASCOTS, **(2009)**.
[10] A. Ban, Flash file system optimized for page-mode flash technologies. U.S. Patent 5,937,425, **(1999)**.
[11] A. Ban, Flash file system. U.S. Patent 5,404,485, **(1995)**.

Authors

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.