Second Chance Replacement Considering a Garbage Collection Cost of FAST Scheme

Ilhoon Shin

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

Abstract

NAND-based storage devices deploy the flash translation layer (FTL) in order to emulate the block device characteristics because NAND flash memory does not support the overwrite operation. The FTL schemes that use log blocks such as the BAST and the FAST scheme are adequate for the devices with harsh memory. This paper presents the log block replacement scheme to improve the performance of the FAST FTL scheme. The presented scheme considers the number of valid pages of the candidate log block when selecting the victim log block, because the cost of the garbage collection decreases as the number of valid pages in the victim log block is less. The presented scheme gives the second chance to the candidate log block if its valid pages are more than the threshold. The simulation shows that the presented scheme improves the performance of the FAST scheme up to 5.0 %. The improvement is more conspicuous as more NAND blocks are used as log blocks.

Keywords: *garbage collection, log block, FAST, flash translation layer, NAND flash memory*

1. Introduction

NAND flash memory is cheap, light, silent, shock-resistant, and energy-efficient. It has been dominating a mobile storage market, and nowadays it is encroaching on the share of hard disk drives in laptops, PCs, and servers markets. The efficient firmware of NAND flash memory, which is called flash translation layer (FTL), has contributed to the success of NAND flash memory.

NAND flash memory is a variant of EEPROM (Electrically Erasable Programmable Read Only Memory) and thus an overwrite function is not supported. The initial state of each cell is clean. Data can be written (programmed) to a clean cell. However, once data are written to a cell, the state of the cell is changed to programmed (dirty), and the programmed cell cannot be re-written. In order to write new data, the cell should be erased, which changes the state of the cell to clean. The worse thing is that the erase unit is larger than the read/write unit.

NAND flash memory consists of blocks. A block is the unit of the erase operation and generally 128 KB or 256 KB in size. A block consists of multiple pages. A page is the unit of the read/write operation and generally 2 KB or 4 KB in size. On the overwrite request, if we erase the block that the target page belongs to and write new data to the target cleaned page, the data of the other pages that belong to the same block will be lost. Thus, we cannot perform an in-place update that overwrites the original page. Instead, an out-of-place update is used. In the out-of-place update, the new data are written to another clean page and the original page is invalidated. The location of valid data becomes different on every re-write request and thus it needs to maintain the mapping information between a logical sector number and its physical location. Performing the out-of-place update and maintaining the mapping information is primary functions of FTL. FTL emulates the block device characteristic that the overwrite operation is supported and makes it possible to use NAND flash memory as storage media of block devices. The efficiency of FTL is one of major factors that determine the overall performance of NAND-based block devices, and thus there were intensive research works to design an efficient FTL [1-5]. However, there is still a room for improvement in the exiting FTL.

The clean pages will be eventually short by the continuous out-of-place update, which initiates a garbage collection process. The garbage collection process selects the victim block for the reclamation of clean pages and moves the valid pages of the victim block to another clean place. Finally, the victim block is erased and the clean pages are reclaimed. Because the garbage collection causes several page copies and block erasures, it is important to reduce both the frequency and the cost of the garbage collection. The aim of this work is to reduce the cost of the garbage collection process with the efficient victim selection scheme.

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 FTL schemes. Section 3 describes the presented victim selection scheme and section 4 evaluates its performance. Finally, section 5 draws a conclusion.

2. Related Work

FTL performs the out-of-place update and maintains the mapping table between a logical sector number and its physical location in order to emulate the block device interface. The mapping unit is a primary factor that determines the performance and the memory requirement of FTL. The previous FTL schemes are classified to the page mapping, the block mapping, and the hybrid mapping FTLs by the mapping unit.

The page mapping FTL [4] maintains the mapping table in a NAND page unit. For this purpose, data are written in a page unit. When modifying the original data, it first finds the clean page and then writes the modified data to the found page. At this time, the original page is invalidated, and the physical location of the target logical page, which is calculated from the logical sector number, is changed to the found clean page. Thus, the mapping table is one-dimensional array, whose index is a logical page number and whose value is a physical page number. On a shortage of clean pages, the garbage collection process is initiated. The drawback of the page mapping scheme is a large memory requirement to maintain the mapping table. The size of the mapping table increases proportionally to the number of NAND pages. Thus, the page mapping FTL is adequate for the block devices that have a large memory such as solid state drives.

The block mapping FTL [5] maintains the mapping table in a NAND block unit. For this purpose, data are written in a block unit. When modifying the original data, it first finds the clean block and then writes the modified data to the found block together with the other unmodified data of the original block. The original block is invalidated and the physical location of the target logical block, which is calculated from the logical sector number, is changed to the found clean block. Thus, the mapping table is one-dimensional array, whose index is a logical block number and whose value is a physical block number. The strength of the block mapping scheme is low memory consumption because the size of the mapping table increases proportionally to the number of NAND blocks. However, its performance is seriously damaged by the excessive coping overhead. Even when modifying the small amount of data, the entire block should be copied to a clean block.

In order to address the drawbacks of the page mapping scheme and the block mapping scheme, the hybrid mapping scheme, BAST (Block Associative Sector Translation) [1] was presented. The BAST scheme uses a portion of total NAND blocks as write buffer. The blocks that are used as write buffer are called log blocks and the other blocks are called data blocks. The data blocks are visible to file systems while the log blocks are invisible. In order to reduce the mapping table size and the copying overhead on the small amount of update, the BAST scheme operates the block mapping scheme for the data blocks and the page mapping scheme for the log blocks. On the write requests, the BAST scheme searches for the associated log block with the target data block, and the data are written to the log block regardless of the logical sector number similarly to the page mapping scheme. If there is no associated log block, a clean log block is allocated and associated with the target data block. The association between the data block and the log block is one to one. The BAST scheme maintains two mapping tables: the block mapping table for the data blocks and the page mapping table for the log blocks. Because a portion of total blocks is used as log blocks, the memory requirement of the BAST scheme is similar to the block mapping scheme. The weakness of the BAST scheme is that it is vulnerable to the small sized random write pattern [2]. By the continuous write requests, the clean log blocks will be eventually short, which initiates the garbage collection. The garbage collection process selects the victim log block and merges it with the associated data block. After merging, the victim log block is reclaimed to a clean log block. The garbage collection process accompanies several page copies and a block erasure, frequent garbage collection damages the performance seriously. However, the small sized random write pattern causes the frequent garbage collection because the log block can be associated with only one data block. The under-utilized log blocks are merged with the data blocks.

In order to address the vulnerability of the BAST scheme against the small sized random write pattern, the FAST (Fully Associative Sector Translation) scheme [2] was presented. The FAST scheme is a kind of the hybrid mapping scheme similarly to the BAST scheme. The difference is that it allows a log block can be associated with multiple data blocks. The FAST scheme manages the log blocks with a FIFO (First In First Out) list. On the write requests, data are written to the current working log block regardless of the logical sector number. If there is no clean page in the current working log block, the next log block in the FIFO list becomes the current working log block. If the next log block does not have clean pages, it is merged with multiple data blocks and reclaimed to a clean log block. The FAST scheme fully utilizes the log block space and thereby reduces the frequency of the garbage collection process. However, the computation overhead of finding the location of the valid sector is considerable because the data of a data block can be distributed to the entire log blocks [3]. However, this computation overhead can be mitigated be the hashed page table that maintains hash lists for the locations of valid sectors in the log blocks [3].

Another restriction of the FAST scheme is that the latency of the garbage collection process can be considerably long if the victim log block has the valid pages of multiple data blocks. In the FAST scheme, the log block is associated with multiple data blocks and thus the log block should be merged with multiple data blocks on the garbage collection, which causes several block erasures and multiple page copies. The long latency of the garbage collection hurts the performance of the FAST scheme and restricts the use of the FAST scheme. This work tries to improve the performance of the FAST scheme by reducing the garbage collection cost.

3. Second Chance Replacement Scheme Considering the Garbage Collection Cost

As described in the section 2, the FAST scheme selects the victim log block in FIFO order, without considering the number of valid pages of the log block. The garbage collection cost increases proportionally to the number of valid pages. For example, if the victim log block does not have valid pages, single erasure is sufficient to reclaim the victim log block. However, if the victim log block has many valid pages, the corresponding data blocks should be merged with the victim log block before erasing the victim log block, which incurs lots of page copies and several block erasures. Thus, it needs to consider the number of valid pages of the log blocks when selecting the victim log block.

In this work, we present a second chance replacement scheme for the FAST scheme. The presented scheme manages the log blocks with the FIFO list similarly to the original FAST scheme. However, when selecting the victim log block, it considers the number of valid pages. If the number of valid pages of the candidate log block is more than the predefined threshold, n, it gets one more chance and the next block in the FIFO list becomes the candidate. As to the next candidate, the same victim selection procedure is performed. If the candidate already got one more chance, it becomes the victim even though it has many valid pages. The chance value of each log block is reset when one of its pages is invalidated by the write requests or by the garbage collection process, because the other valid pages are also likely to be invalidated in a short time.

4. Performance Evaluation

In order to evaluate the effect of the presented second chance replacement scheme, we compared its performance with the original FAST scheme that uses the FIFO replacement. In the both schemes, the hashed page table [3] was used to reduce the computation overhead. In the second chance replacement scheme the threshold that determines the victim log block was fixed to 0 because only single erasure of the victim log block is done on the garbage collection. The performance of each scheme was measured with the simulator that models NAND-based storage. The simulator assumes that the latencies of read, write, and erase operations of a NAND page and a NAND block are 25 us, 200 us, 2 ms, respectively and that the NAND based storage organizes multiple NAND flash memory chips in 2-channel & 4-way structure. A physical NAND page is 2KB in size, and a physical block is 128 KB in size. Transferring 2KB data via a channel is assumed to take 70 us. The performance measure of each scheme is the total I/O time, which is calculated using the following formula: (total elapsed time = page read count × page read latency + page write count × page write latency + block erase count × block erase latency) [2, 3]. Two realistic workloads that were collected in a PC environment while performing internet browsing, document editing, and so on were used. The partitions were formatted with NTFS file system. The partition size of the NTFS1 trace was 32 GB, and the partition size of the NTFS2 trace was 80 GB.

Figure 1 shows the result. The x-axis denotes the log block ratio, which is varied from 1 % to 9 % of the total NAND blocks. The y-axis denotes the total I/O time in seconds. FAST denotes the original FAST scheme, and FAST-SC denotes the FAST scheme that deploys the presented second chance replacement scheme. The result shows that the second chance replacement scheme that considers the number of valid pages of the log blocks is effective in all the configurations considered in the both traces. The improvement increases as more NAND blocks are used as the log blocks. When the log block ratio is 9 %, the performance improvement is about 4.4 % in NTFS1 and about 5.0 % in NTFS2 trace.

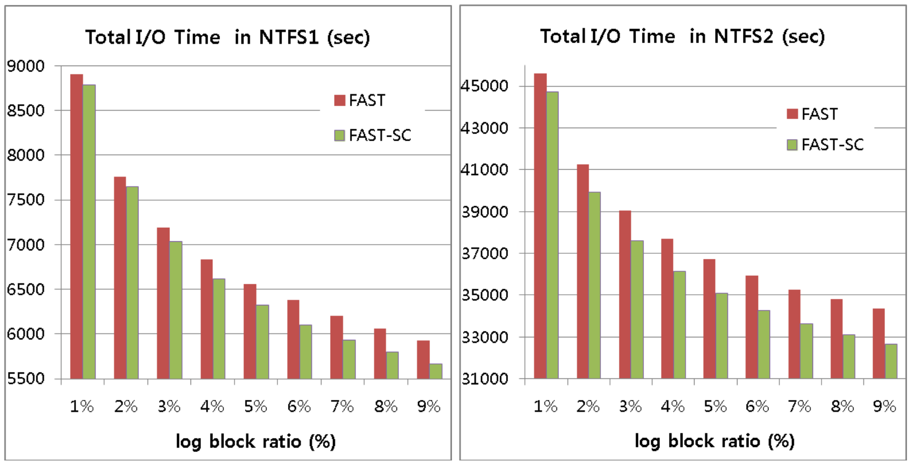

Figure 1. Elapsed Time Varying Log Block Ratio (NTFS1 & NTFS2)

5. Conclusion

In this paper, we presented a new replacement policy of log block for the FAST FTL scheme. When selecting the victim log block on the garbage collection, the presented scheme considered the number of valid pages of the candidate log block. If the number of valid pages was more than the predefined threshold, it got another chance, and the next block in the FIFO list became the candidate. If the candidate already had got another change, it became the victim regardless of the number of valid pages. In the simulation, the predefined threshold was fixed to 0 because single erasure of the victim is sufficient if the victim log block is fully invalidated. The simulation result showed that the presented replacement scheme was effective in all the configurations considered. The improvement increased as more NAND blocks were used as log blocks. In the considered configurations, the maximum improvement was about 5.0 %.

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] 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)**.

[2] 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)**.

[3] I. Shin, “Reducing computational overhead of flash translation layer with hashed page tables”, IEEE Transactions on Consumer Electronics. Vol. 56, pp. 2344-2349 **(2010)**.

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

[5] 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.