FTL Algorithm using Warm Block Technique for QLC+SLC Hybrid NAND Flash Memory

Wanil Kim[1], Seok-Bin Seo[2], Jin-Young Kim[3], Se Jin Kwon[4]
Department of Computer Engineering
Kangwon National University
Samcheok, South Korea

***Abstract*—When applying the existing flash translation layer technique to a mixed NAND flash storage device composed of Quad Level Cell and Single Level Cell, because the characteristics of a semiconductor chip are not taken into consideration, the data are stored indiscriminately, and thus the performance and stability are not guaranteed. Therefore, this study proposes a flash translation layer algorithm using the warm block technique in a NAND flash storage device that combines a large capacity Quad Level Cell and a high performance Single Level Cell. The warm block technique avoids overloading of the read/write/erase operations in the Quad Level Cell flash memory by efficiently placing hot data that are frequently updated on a long-living Single Level Cell. It was confirmed experimentally that the lifetime extension and performance of hybrid NAND flash memory are improved using the warm block technique.**

Keywords*—Quad level cell; single level cell; composed flash memory; flash translation layer

I. Introduction

Flash memory is classified into various types of semiconductor chips including Single Level Cell (SLC), Multi- Level Cell (MLC), Triple Level Cell (TLC), and Quad Level Cell (QLC) depending on the number of bits that can be stored in a single memory cell. SLC, MLC, and TLC/QLC are treated as advanced, intermediate, and TLC/QLC entry-level types, respectively. A hybrid SSD, in which various types of chips are mixed and used in a single storage device, has recently been proposed [1]. Because such a storage device has high-grade and intermediate/entry-level semiconductor chips in a single SSD, it is necessary to decide where to store the data, namely, in either the high-grade or intermediate/low-cost semiconductor chips when a write request is made from the file system. If the data classification and storage technique are inefficient, unnecessary data movement from a merge operation frequently occurs, which may degrade the overall performance.

To improve this, studies on the flash translation layer (FTL) [2] have been carried out. Most existing approaches [3] write updates to the log block. However, when such a log block-based FTL is applied to hybrid NAND flash memory [4], the characteristics of each semiconductor chip are not considered and data transfers between the high-grade and intermediate/low-cost semiconductor chips frequently occur, resulting in a degraded performance.

The algorithms in this paper are designed to minimize the erase operation of intermediate/supplied semiconductor chips and increase I/O performance of high-end semiconductor chips, which are for large storage devices based on QLC+SLC mixed NAND flash.

Section 2 describes related studies and their background. Section 3 provides the structure of the proposed method. Section 4 details the operation of the proposed method. Section 5 shows the performance test results of the proposed method as compared with the existing techniques. Finally, some concluding remarks are provided in Section 6.

II. Related Studies and their Background

Previous studies on hybrid NAND flash memories [5] have focused on the development of cold data storage on a relatively low-performance chip (TLC, MLC) depending on the characteristics of the flash memory chip, and on storing hot data on a high-performance chip (SLC).

In an existing study, where mixed NAND flash memory is applied with the technology of FAST [6], MLC is used as a data block and SLC is used as a log block. Figure 1 shows the structure of the existing technique [7]. This technique is written to the log block when the data are updated, and when the log area is full, the data blocks associated with the log block are fully merged [8], resulting in a degraded performance. Partial merges, which are relatively low in terms of computational cost, are applied in existing studies only when the sequential write condition of the data block is satisfied.

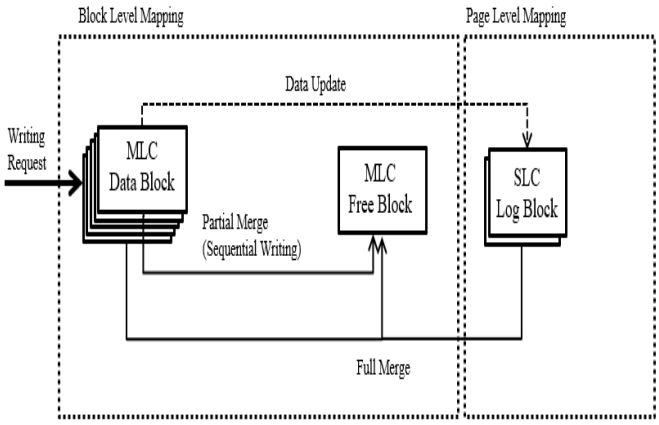

Fig. 1. Existing Log-Block based Mixed NAND Flash Storage Architecture.

Therefore, full merge operations are frequently conducted, meaning that the frequently used data in the log block are handled as cold data indiscriminately and written as MLC data block. When a conventional log block FTL technique is applied to hybrid NAND flash memory, the characteristics of a semiconductor chip are not considered, the data are indiscriminately written, and the performance and stability are thus not guaranteed. With this background, the present study proposes FTL algorithm based on the warm block technique in a NAND flash storage system that combines a high-capacity QLC and a high-performance SLC when considering the characteristics of a semiconductor chip.

III. Structure of the Proposed Warm Block Technique

This study uses large QLC NAND flash as both data and warm blocks, and uses SLC NAND flash as a log block. The proposed method uses the warm block technique with a log-block based algorithm, and has four types of physical blocks namely, a data block, warm block, hot data block, and free block.

In a conventional FTL [9], 3–5% of the data blocks are fixedly allocated as log blocks; however, this study proposes the allocation of 3% log blocks and 2% warm blocks.

The warm block exists between the data and log blocks. On the other hand, in the existing log-block based FTL algorithm, the updated data are written to the log block immediately when the data in the data block are updated. The proposed method regards the data updated from the data block as warm data before writing to the log block, and stores the updated data according to the offset after the warm block allocation of the QLC.

When the data in the data block are updated, as shown in Figure 2, the updated data are written to the warm block according to the offset of the corresponding logical address.

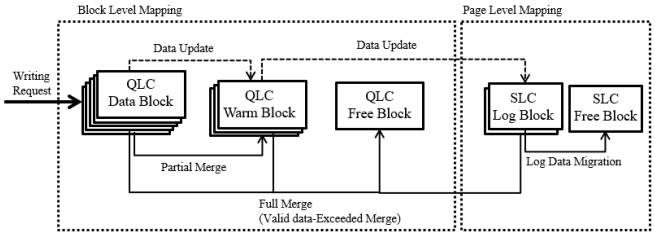

Fig. 2. Mixed NAND Flash Storage Architecture Applied using the Proposed.

This implies that partial merges will be possible in the future regardless of whether the write requests are sequential or arbitrary. Therefore, unlike the existing technique, which can apply partial merge only when the sequential writing condition is satisfied, the proposed technique can be partially merged between the data and warm blocks under any situation.

When the data in the warm block are updated, the data are treated as hot data, and the updated data are written in the SLC log block. The data treated as hot data are maintained in the SLC log block for a long period of time. This occurs because the hot data are located in the log block as long as possible through a self-applied garbage collection in the log area owing to the presence of invalid data in the log block.

If there are no invalid data in the log area, then a full merge operation will take place only then. This means that all log blocks in the log area are filled with valid data, indicating that the space utilization of the log area has already been maximized.

The merge operations of the proposed technique are explained in detail in Section 4.

IV. Operation of the Proposed Warm Block Technique

A. Data-Warm Partial Merge

If the data in a data block are updated, and an allocation to a warm block is required but there are no more free blocks to allocate, Data-Warm partial merge is conducted by selecting the block with the smallest merge operation cost as a victim block. If there are no valid data in the selected victim block, it is possible to execute a switch merge that has the least computation cost during the merge operation.

Switch merges and partial merges of the log-block based existing techniques can be conducted only on sequential data of log blocks handled by page mapping; however, because the warm block data are stored at a position depending on the offset regardless of the writing request type, sequential or random, the proposed method can switch and partially merge under any type of situation.

The Data-Warm partial merge in Figure 3 shows that the D1 and F1 data in the data block apply a “copy and write” operation using the D1' and F1' data of the warm block. Next, the warm block is replaced with the data block after the erase operation of the data block occurs. In this case, the erase operation occurs only once.

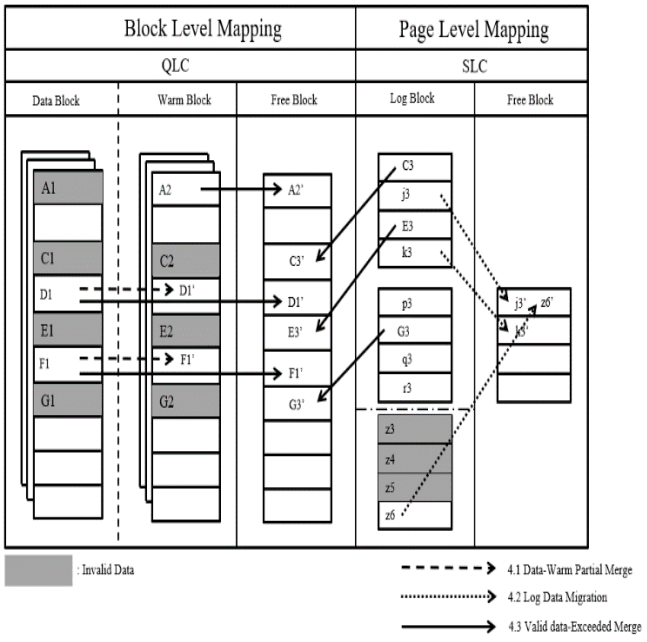

Fig. 3. Three Merge Operations of the Proposed Technique.

B. Log Data Migration

If the SLC log blocks are filled with data, a block with a large amount of invalid data is selected as a victim block, and a valid data migration, which is moving data from a victim block to a free block, is conducted. As shown in Figure 3, the block in which the z3, z4, z5, and z6 data are stored within the log block is selected as a victim block because it has the largest amount of invalid data. Because the block whose valid data in the victim block is z6 data has the greatest amount of invalid data, it is selected as the victim block, and the Log Data Migration is performed from the victim block to a free block as moving valid z6 data. After conducting the data migration, the victim block is erased to make the next available free space. This results in only one erase operation.

In contrast, after assuming that there are no blocks in which z3, z4, z5, and z6 data are stored, all of the log blocks are stored as valid data. In this case, if the data in the worm block is to be written to the log block at a request to be updated, a Valid Data-Exceeded Merge operation occurs. The Valid Data-Exceeded Merge operation is discussed in Section 4.3. The j3 and k3 data remaining after the Valid Data-Exceeded Merge are positioned as doing the Log Data Migration into a free block.

C. Valid Data-Exceeded Merge

This is a full merge of data blocks, warm blocks, and log blocks, which is performed when log blocks are filled with valid data. A block with the lowest degree of association among the log blocks is selected as a victim block, and the Valid Data-Exceeded Merge is performed on the data block and warm block having the largest amount of data included in one of the LBNs.

However, if data are included in the LBN other than the victim block, as shown in G3 of Figure 3, the data are only migrated to the free block of the QLC, but the log block does not execute the Log Data Migration. Only one log block selected as a victim block conducts the Log Data Migration, and the victim block is erased to make the next free space available. This is because what is needed in that situation is only one block, and is also to avoid the problem of further updates if there is data in a location other than the corresponding LBN-consistent victim block.

In this case, an erase operation is applied three times, namely, to a data block, a warm block, and a log block. Valid Data-Exceeded Merge occurs when the entire log block is filled with valid data, which means that the space utilization of the log area has already been maximized.

D. Write operation

Algorithm 1 describes the write operation algorithm of the proposed scheme. The empty space in the offset of the data block corresponding to the input logical page number is checked, and if empty, the data are written in the page of the corresponding offset (lines 1 and 3). If the page of the offset is not empty, the presence or absence of the assigned warm block should be identified.

If there is a warm block already allocated to the data block, the page space of the corresponding offset of the warm block is checked; if empty, the data are written into the corresponding warm block (line 7), but if not, the logical page number is considered hot and written on the log block.

Here, if there is a blank page space in the log block, the data can be written on the page (line 10). If there is no empty page space in the log block, a free space needs to be secured in the log area. To do so, the presence of invalid data in the entire log block is first identified; if invalid data exist, after Log Data Migration and one erase operation, log data are written into the empty space secured (line 13). However, if all log block data are full of valid data, because it is impossible to secure a free space in the log area, then garbage collection in the related data block, warm block, and log block is conducted by Valid Data-Exceeded Merge to occupy free space to write new data (line 14). If there is no warm block assigned to the data block, the warm block that can be allocated in the warm area is inspected. If an allocable warm block exists, data are written after allocation (line 20). If no warm blocks are available, Data-Warm partial merge is conducted to execute the garbage collection and secure the assignable warm block. The generated warm block is assigned to the corresponding data block and the data are written (line 21).

ALGORITHM 1 WRITE OPERATION

```
1:  Input: Logic Page Number (LPN), Data
2:  if In Data block, The page of the corresponding LBN's offset is empty then
3:      Write on the corresponding page;
4:  else
5:      if Warm block is already allocated to data block then
6:          if In Warm block, corresponding page of the offset is empty then
7:              Write data in the page of the corresponding offset page;
8:          else
9:              if In Log block, a blank page exists then
10:                 Write data in the log block's blank page;
11:             else
12:                 if In Log block, a invalid data exists then
13:                     Write data after execution of Log Data Migration
                            and erase the invalid block;
14:                 else Write data after execution of Valid Data-Exceeded Merge
                            and erase the invalid blocks;
15:                 end if
16:             end if
17:         end if
18:     else
19:         if Allocable Warm block is available then
20:         Write after Warm block allocation;
21:         else Write after Data-Warm partial merge;
22:         end if
23:     end if
24: end if
```

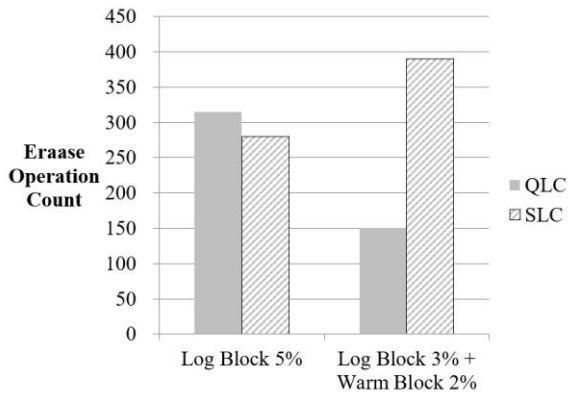

Fig. 4. Comparison of Erase Operations before and after Applying Warm Block Technique.

V. Conclusion and Future Plans

In this study, we proposed an algorithm for minimizing the erase operations of a QLC semiconductor chip and increasing the input/output performance of an SLC semiconductor chip by utilizing the warm block technique in the mass storage system, which is a mixture of QLC and SLC.

To measure the performance of QLC + SLC mixed NAND flash memory, QLC and SLC hardware characteristics were implemented based on existing data sheets in [10] and [11], and a performance evaluation was conducted after the warm block technique was implemented in the device drive of mixed NAND flash memory. In this study, the data sent from the file system were modified to generate about 10,000 write requests to improve an intuitive understanding.

Figure 4 shows the number of operations conducted in QLC + SLC mixed NAND flash memory when a conventional log block algorithm and the warm block technique are applied. Although the total numbers of erase operations in both the existing algorithm and the warm block algorithm are similar, the number of erase operations conducted in the QLC and SLC chips is quite different. A more intensive erase operation is conducted on a QLC chip for the existing log block algorithm (QLC 315, SLC 281), whereas an SLC chip is more intensely used for the warm block technique (QLC 153, SLC 392).

These experimental results support the idea that the warm block technique can improve the overall lifetime of QLC + SLC mixed NAND flash memory.

Considering the characteristics of the semiconductor chip mentioned in Section 1, the lifetime of QLC flash memory is 1,000 operations or less, whereas the lifetime of SLC flash memory is 100,000 operations or more [10], [11]. Therefore, in this study, an overload of read/write/erase operations in QLC flash memory can be prevented by efficiently allocating hot data with frequent updates to a long-lived SLC. In other words, the lifetime extension and performance of QLC + SLC hybrid NAND flash memory are improved using the warm block technique, as proposed in this study.

Future studies will apply the warm block technique to various FTL algorithms in addition to the log block algorithm, and we plan to propose a wear-leveling algorithm that optimizes the compatibility with the warm block technique. In addition, based on actual workloads used in smartphones and servers, we plan to compare the performance with existing techniques, including the number of erase operations and the operation speed of the write and read requests in various environments.

Acknowledgment

This work was supported by Basic Science Research through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A3B04031440). This study was also supported by 2018 Research Grant from Kangwon National University (No. 000000000).

References

[1] S. Hong and D. Shin, "NAND Flash-Based Disk Cache Using SLC/MLC Combined Flash Memory," 2010 International Workshop on Storage Network Architecture and Parallel I/Os, Incline Village, NV, pp. 21-30, 2010.

[2] Dongzhe Ma, Jianhua Feng, and Guoliang Li, “A survey of address translation technologies for flash memories,” ACM Computing Surveys (CSUR), 46(3), pp. 1-39, 2014.

[3] Jesung Kim, Jong Min Kim, S. H. Noh, Sang Lyul Min and Yookun Cho, "A space-efficient flash translation layer for CompactFlash systems," in IEEE Transactions on Consumer Electronics, vol. 48, no. 2, pp. 366-375, May 2002.

[4] Rino Micheloni, “Solid-state drive (SSD): A nonvolatile storage system,” Proceedings of IEEE 105(4), pp. 583-588, 2017.

[5] Se Jin Kwon and Tae-Sun Chung, “Data pattern aware FTL for SLC+MLC hybrid SSD,” Design Automation for Embedded Systems 19(1-2), pp. 101-127, 2015.

[6] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park, and Ha-Joo Song, “A log buffer-based flash translation layer using fully-associative sector translation,” ACM Transactions on Embedded Computing Systems (TECS) 6(3), no. 18, pp. 1-27, 2007.

[7] Byung-Woo Nam, Gap-Joo Na, and Sang-Won Lee, “A hybrid flash memory SSD scheme for enterprise database applications,” Web Conference (APWEB), 2010 12th International Asia-Pacific, IEEE, 2010.

[8] Lee, Sang-Won, et al. "A log buffer-based flash translation layer using fully-associative sector translation." ACM Transactions on Embedded Computing Systems (TECS) 6.3 (2007): 18.

[9] Gupta, Aayush, Youngjae Kim, and Bhuvan Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. Vol. 44. No. 3. ACM, 2009.

[10] Toshiba Semiconductor, “BiCS3 768 Gb 3D QLC NAND chips,” Toshiba Semiconductor Technical Notes, 2018.

[11] Micron Electronics, “MT29E128G08CECAB, MT29E256G08CMCAB, MT29E512G08CUCAB,” Micron Electronics Datasheet, 2018.