Turkish Journal of Computer and Mathematics Education Vol.12 No.13 (2021), 2815-2822

Research Article

A Block Classification Method with Monitor and Restriction in NAND Flash memory

Myungsub Lee

Department of Computer Information, Yeungnam University College, Daegu, Republic of Korea
skydream@ync.ac.kr

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 4 June 2021

Abstract*:*Flash memory has the advantages of fast access speed, low power consumption, and low price. Therefore, it is widely used in many sectors of the electronics industry. However, the flash memory also has a disadvantage of a limited number of P/E (Program/Erase) cycles. Many wear-leveling techniques have been studied to prolong the lifetime of flash memory by equalizing the P/E cycles of the blocks. This paper proposes BCMR (a Block Classification with Monitor and Restriction) method to isolate and reduce the interference of blocks in garbage collection and wear leveling. The BCMR method monitors the endurance variation of blocks during garbage collection and detects hot blocks by making a restriction condition based on this information. This method induces block classification by its update frequency for garbage collection and wear leveling, resulting in a prolonged lifespan for NAND (Not AND) flash memory systems. This paper shows the block standard deviation, which was reduced by up to 11% when BCMR was implemented. And we show the average number of page migrations during garbage collection and wear leveling, which was reduced by about 1% when BCMR was activated. The performance evaluation results show that the BCMR method prolonged the life of NAND flash memory systems by 3.95% and reduced the standard deviation per block by 7.4%, on average.

Keywords:monitor and restriction, garbage collection and wear leveling, lifespan, block erase count, block standard deviation

1. Introduction

NAND (Not AND) flash memory can be accessed by page or block units grouped in a series, providing a relatively high level of directness. As a result, NAND flash memory is not only used in portable devices, such as USB (Universal Serial Bus) drives, SD (Secure Digital) cards, and smart phones, but also has a wide variety of uses such as in SSD (Solid State Drive) backup storage devices. Despite this advantage, NAND flash memory has a shorter lifetime than the existing backup storage devices because of limited overwriting in the same location. To resolve this problem, many researchers have presented techniques that consider NAND flash memory's special characteristics, for example, the technique for extending NAND memory's lifetime is called wear leveling [1, 5].

In wear leveling, data separating blocks into "cold" and "hot" according to the number of erasures is saved in a table along with various other information, such as page reference counts, reference, and erasure time; this is used to extend the flash memory's lifetime. However, if this data is saved in memory, this introduces the disadvantage of the increase in memory usage, and the implementation of embedded systems with low specifications becomes difficult. Therefore, researchers are exploring wear-leveling techniques that use a BET (Block Erase Table), which is arranged by bits and can reduce memory usage [2-4]. As a BET's k value increases, multiple blocks are bundled into a single group to correspond to each bit. Therefore, as the k value increases, memory usage could be further reduced. However, as the k value increases, it seriously limits the performance of wear leveling.

To overcome this disadvantage in BET techniques, the hidden cold block problem has been identified as a cause of reduced performance, and to resolve this, wear-leveling techniques, which use bit arrangements and BST (Bit Set Thresholds), have been proposed [3]. To resolve the hidden cold block problem, BST determines if the bit mapped to the block group, in which the erasure operation occurs, has been set to 1. Moreover, if the block group's average number of invalid pages is greater than the overall block group's average number of invalid pages, the BET bit is set to 2, thus resolving the hidden cold block problem due to an increase in the k value. However, BST is only effective if $k \geq 1$, which limits the performance increase because of BET.

To resolve these problems, this paper proposes the tracking cold blocks technique, which strengthens the extent to which cold blocks are recognized and minimizes the migration of pages due to wear leveling to reduce the overhead and improve the overall memory lifetime.

Unlike magnetic disks, NAND flash memory has problems such as a shorter lifespan, an erase-before-write requirement, and imbalance of operation units [1-4]. To address these problems, many researchers have proposed file systems or data structures considering the inherent characteristics of NAND flash memory [6-8]. However, recently considerable research has been conducted on FTL (Flash Translation Layers), a development that can solve these problems without changing the traditional file systems or data structures.

FTL is composed of an address translation table, garbage collection, and wear leveling [7]. Previous researchers have conducted studies on methods to extend the lifespan of NAND flash memory using either garbage collection or wear leveling. However, these researchers have proposed optimization methods to extend life assuming that each component of the FTL layer operates alone. Therefore, problems arise when certain blocks are concurrently

selected in the system environment where the proposed methods are simultaneously connected and performed in the FTL layer. This causes shortening of the NAND flash memory lifespan. Therefore, in system environments with FTL layers, in which garbage collection and wear leveling are simultaneously considered, the problem of wear in certain blocks can be solved easily by classifying and allocating separate blocks to the two processes.

This paper proposes a block classification with monitor and restriction methodology to classify frequently selected blocks in a system environment in which garbage collection and wear leveling are simultaneously conducted. The proposed methodology uses the following procedure in the process of garbage collection.

(1) Information on changes in the block erase count is monitored.
(2) The sacrifice block is selected based on the monitored information.
(3) When the sacrifice block is selected, the constraints are applied.

Based on the above procedure, frequently selected blocks are selected according to the algorithm proposed in this paper, thereby helping extend the lifespan of NAND flash memory. This paper is structured as follows. Section 2 discusses the background information and related research on NAND flash memory. Section 3 details the process of performing BCMR. Section 4 describes a performance evaluation of BCMR, and Section 5 concludes the paper.

2. Related Work

As shown in Table 1, SLC(single-level cells), MLC(multi-level cells), or TLC(triple-level cells) are implemented in NAND flash memory depending on the method in which bits are classified in flash cells [8].

Table 1. Comparison of flash cells

Description	SLC	MLC	TLC
Read latency	25 μs	60 μs	100 μs
Program latency	200 μs	800 μs	2.4 ms
Erase latency	700 μs	1.5 ms	3.0 ms
Bits per flash cell	1 bit	2 bit	3 bit
Endurance	10^5	10^4	10^3
Price per bit	high	low	very low

As shown in Table 1, as the number of bits per flash cell in NAND flash memory increases, operation latency increases. Because an increase in bits per flash cell has the effect of decreasing price per bit, recently, MLC has mainly been used for its trade-off between performance and price. In addition, NAND flash memory has inherent characteristics different from conventional magnetic disks, which are described as follows [1-2].

(1) NAND flash memory operations include read, write, and delete. Read and write operations are executed by page units, whereas delete and block operations are executed by block units. Here, a block is a group of multiple pages.

(2) As shown in Table 1, the latencies of read, write, and delete operations of NAND flash memory are asymmetric.

(3) Unlike magnetic disks, there is a limited number of write cycles per block in NAND flash memory.

To address the above-mentioned problems of NAND flash memory, initial studies have proposed specialized file systems or data structures; however, the recent studies have mainly used the FTL, a method that supports the properties of magnetic disks as-is. The structure of FTL is shown in Fig. 1.

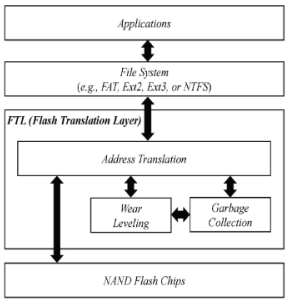

Fig. 1. System architecture of FTL

The address translation table records the logical and physical addresses as one item; accessing the physical address in NAND flash memory is supported by using the logical address. Through this method, NAND flash memory overcomes the problems of existing operations that execute delete operations before write operations. In

other words, in-place update of NAND flash memory is possible by using the address translation table. However, a number of invalid pages arise when this process is repeated, thereby taking up unnecessary memory space. Therefore, to clean up this unnecessary space, a garbage collection process is needed.

Garbage collection supports the function of collecting the invalid pages of NAND flash memory, which is carried out during the in-place update support or data deletion. However, if garbage collection is executed repeatedly, a problem arises in which only certain blocks frequently execute delete operations, thus becoming the main cause for NAND flash memory life degradation. To address this issue, the wear leveling method is used.

Wear leveling is a technique to address the problem of imbalanced delete operations between blocks that arise in garbage collection. The number of delete operations on each block is equalized to extend the lifespan of NAND flash memory. In other words, wear leveling plays a role in leveling the wear of memory by selecting blocks that have almost no delete operations to participate in garbage collection. Most recent research on methods to extend the life of NAND flash memory uses the afore-mentioned garbage collection and wear leveling techniques [10].

Initial research on extending the life of NAND flash memory focused on securing space for use. As a result, certain blocks in memory were found to be frequently used. This practice was found to deplete NAND flash memory life faster than expected. To address this problem, the CB, CAT, and SAGC methods have been proposed [11].

Recently, wear leveling has been actively researched to extend the life of NAND flash memory. A representative study is SWL. In SWL, a bit flag table is used to identify the blocks that have not participated in garbage collection and make them to participate in garbage collection, thereby leveling the memory wear and extending the life of NAND flash memory.

However, the proposed methods are efficient when either garbage collection or wear leveling is used and do not consider the system environment that uses both processes simultaneously. For example, to employ wear leveling efficiently, only blocks that contain pages that have almost no write operations should be selected to participate in garbage collection. To achieve this, the pages that participate and those that do not participate in the operation should be classified and transferred to separate blocks. As a result, the migrated pages are situated between the hot block (block that frequently deletes) and the cold block (block that frequently does not delete).

In a system that considers both garbage collection and wear leveling, there is a risk that the blocks between the hot and cold blocks are frequently referenced. Therefore, in a system environment in which garbage collection and wear leveling are performed simultaneously, it is necessary to separate the hot and cold blocks. To fulfil this requirement, this paper focuses on preventing interference and ensuring isolation of the selected blocks between the two processes.

3. Proposed Method

This paper proposes a block classification with monitor and restriction method to prevent interference of blocks and ensure isolation in the garbage collection and wear leveling process. As shown in Fig. 2, the main procedure of the BCMR method is executed in the following three steps.

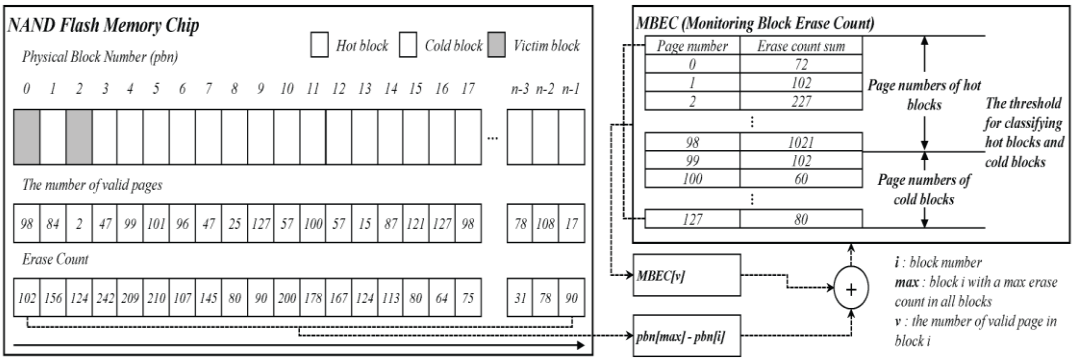

Fig. 2. Procedure of BCMR

As shown in Fig. 2, the main procedure of the BCMR method is executed in the following three steps

(1) In the garbage collection execution process of BCMR, all blocks are monitored to measure changes in the block erase count. Based on this information, when the hot block is being wear leveled in garbage collection, the cold block is selected. Here, a hot block refers to the block in which write operations occur frequently, whereas a cold block refers to the block in which write operations occur rarely.

(2) Based on the measured changes in the erase count, cold blocks are excluded from selection in the garbage collection process.

(3) Finally, in the wear leveling process, the hot blocks and cold blocks are classified based on the monitored changes in the erase count in garbage collection and a new threshold is set. Through this process, interference is prevented, and isolation is ensured between the two processes.

As shown in Fig. 2, the MBEC (Monitoring Block Erase count) table is used to measure changes in the block erase count during the execution of garbage collection. The valid page numbers of the frequently selected blocks can be determined by using the information recorded in the MBEC table. Thus, BCMR can measure the block erase imbalance by using the MBEC table to track differences in the changes in the erase count of the blocks preferred by garbage collection. In addition, BCMR also clearly classifies the status of the corresponding block as a hot or cold block to determine the number of valid pages.

The details of the BCMR procedure shown in Fig. 2 are given in Algorithm 1.

Algorithm 1: BCMR procedure

```
    input: max_ec, max_blk, ec, mbec, v_pages, cost, i, and v_th
    output: null
        // Get the maximum number of erase count
1       max_ec ← get_max_erase_count();
2       for i = 0 to max_blk do
            // Measure a difference between a maximum erase count and a corresponding erase count.
3           mbec[v_pages[i]] = max_ec - ec[i];
            // Ignore an corresponding block i when it satisfies this condition.
4           if v_pages[i] > v_th then
5               continue;
6           end
            // Estimate a cost of block i on each technique.
7       end
        // Reclaim victim blocks by garbage collection.
8       if wear_leveling_condition = true do
            // Get a bcmr cost by equation (1).
9           cost ← get_bcmr_cost();
10          if cost > 0 then
11              v_th = v_th - 1;
12          else
13              v_th = v_th + 1;
14          end
            // Reset the mbec table and perform procedures in each technique.
15      end
```

Through Algorithm 1, BCMR monitors changes in the erase count for all blocks and uses this information to classify hot blocks and cold blocks. In wear leveling, Eq. 1 is used to reset the threshold to prevent the blocks not the target of garbage collection from participating in garbage collection.

$$\text{cost} = \sum_{i=0}^{T} BCMR(i) - \sum_{i=T}^{max} (BCMR(i)\ x\ r)$$

In this equation, i indicates page number, T indicates the threshold that classifies the existing hot blocks and cold blocks, max indicates the maximum number of pages, and r indicates the weight to be assigned to the cold blocks. Based on the cost obtained from Eq. 1, the new threshold of BCMR is calculated as follows.

(1) If cost exceeds 0, then the change in the erase count of the hot block is large and threshold value is accordingly reduced.

(2) If cost is less than 0, then the change in the erase count of the cold block is large and the threshold is accordingly increased.

In BCMR, alongside the existing ET (Erase Table), an additional table is used that memorizes moved blocks judged to be cold blocks. This is called the MCT (Migrated Cold block Table). For wear leveling, BCMR targets the blocks in which all the ET and MCT bits are zero to prevent indiscriminate page movement and excludes unsuitable blocks from garbage collection to improve NAND flash memory performance.

Algorithm 2 describes the detailed process by which BCMR uses ET and MCT. According to algorithm 2, after ga_{victim} garbage collection is performed, if the total block erasure count e_{cnt} divided by the erased block group count f_{cnt} is greater than the threshold value T, wear leveling is performed. If all the block groups mapped to ET or MCT are erased, e_{cnt}, f_{cnt}, ET, and MCT are all reset to 0 (Algorithm 2; Steps 3–13).

Algorithm 2: BCMR Static Wear Leveling Procedure

```
   input: e_cnt, f_cnt, bcmr_cnt, k, f_index, ga_victim, bcmr_vimtim, ET, MCT, and T
   output: null
1  ga_victim ← GetVictimBlockByGreedy();
2  if e_cnt/f_cnt ≥ T do
3    if f_cnt ≥ size(ET) then
4      e_cnt ← 0;
5      f_cnt ← 0;
6      ResetAllFalgsInET();
7      return;
8    end
9    if bcmr_cnt ≥ size(MCT) then
10     bcmr_cnt ← 0;
11     ResetAllFalgsInBCMR();
12     return;
13   end
14   for f_index = 0 to size(ET) do
15     if ET[f_index] = 0 and MCT[f_index] = 0 then
16     end
17   end
18   for ga_victim mod k to (ga_victim mod k) + k do
19     if IsErasedBlock(ga_victim) = false then
20       EraseBlock(ga_victim);
21     end
       //migrate pages in bcmr_victim to ga_victim
22     MigrateTo(ga_victim, bcmr_victim);
23     EraseBlock(bcmr_victim);
24     MCT[idx(ga_vimtim)] ← 1;
25   End
26   bcmr_cnt ← bcmr_cnt + 1;
27 end
```

If the above scenario does not apply, that is, if wear leveling should be performed, the $bcmr_{victim}$ block group, which will be migrated, is selected according to Eq. (2) from among the block groups with ET and MCT bits, which are all zero for the same index (Algorithm 2; steps 14–17).

In Eq. 2, v_i is the group's i-th block's valid page ratio; the group with the lowest *cost* is selected to be $bcmr_{victim}$.

$$\text{cost} = \sum_{i=0}^{2^k} 1 - v_1 \tag{2}$$

If $bcmr_{victim}$ is selected, its data is migrated to the block group in which garbage collection was most recently performed (Algorithm 2; steps 18–25). Then, $bcmr_{victim}$ is erased, and the mapped ET bit is set to 1 (Algorithm 2; step 20).

Simultaneously, to represent the migrated block group determined to comprise cold blocks, the MCT bit mapped to ga_{victim} is set to 1, and the cold block group count $bcmr_{cnt}$ is incremented (Algorithm 2; step 26).

Algorithm 3 describes the ET update process via erasure.

Algorithm 3: ET Update

```
   input: e_cnt, f_cnt, k, b_index, and ET
   output: e_cnt, f_cnt
```

/* *ET* are updated based on b_{index} and *k* in the ET mapping when the block is erased. */

1 $e_{cnt} \leftarrow e_{cnt} + 1$; // *Increase the total erase count.*

// Update the ET if needed

2 **if** $ET[b_{index}/2^k] = 0$ **do**

3 $ET[b_{index}/2^k] \leftarrow 1$;

4 $f_{cnt} \leftarrow f_{cnt} + 1$;

5 **end**

In Algorithm 3, the ET update occurs if erasure occurs because of garbage collection or wear leveling. If an erasure occurs on a block of a group, the ET bit mapped to the block group's index is calculated and the ET bit is set to 1. In addition, the overall block erasure count e_{cnt} and the erased block count f_{cnt} are incremented (Algorithm 3; steps 1–5).

It considers blocks in which garbage collection has been performed as hot blocks, and it considers blocks in which the ET and MCT bits are zero, that is, block groups in which erasure has not been performed and the valid page ratio is high to be cold blocks. As a result, BCMR detects hot blocks through garbage collection and uses ET and MCT to strengthen cold block detection. This prevents indiscriminate page migration, and excludes cold blocks from participating in hot blocks' frequent garbage collection to the greatest extent possible. Thus, the garbage collection efficiency is improved and the overhead associated with the page migration policy is reduced. We can expect that the improved wear leveling will increase the NAND flash memory's lifetime and improve its overall performance.

As a result, the threshold of BCMR classifies hot blocks and cold blocks on the basis of the difference between the block erase counts, which is the information required during the garbage collection process to prevent interference and ensure isolation of the selected blocks between the two processes.

4. Experiment and Evaluation

To evaluate the performance of the proposed BCMR method, we performed simulated experiments by using DiskSim (Extension for SSD), which is based on DiskSim 4.0; SPC (Storage Performance Council) of UMass Trace Repository and FIU Filesystem System Call Traces of IOTTA Repository were used as the workload. SPC was set to include the characteristics of the application performed in OLTP (On-Line Transaction Processing) and FIU was set to include operational characteristics such as terminal users and web services.

The performance evaluation of BCMR was based on data such as NAND flash memory life, valid page migration, and block standard deviation. First, for the memory life and block standard deviation, the erase count for each block was set to 10^3, and the block standard deviation was set based on the block time that first exhausted all erase counts among all blocks. A comparative analysis was conducted with the valid page migration value based on the number of page migrations after 100 million write operations for overheads. The experimental results in Fig. 3 show the NAND flash memory life.

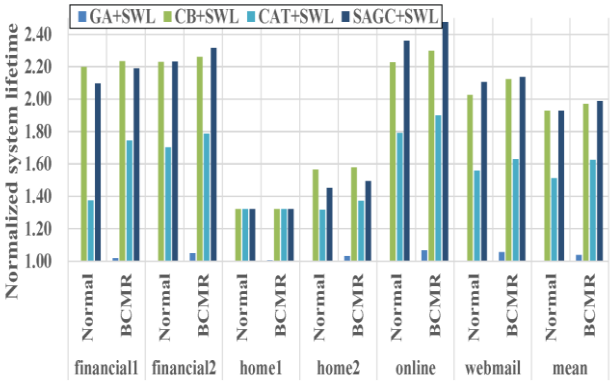

Fig. 3. The lifespan of NAND flash memory system

As shown in Fig. 3, the average lifespan of NAND flash memory was extended by 3.58%, 2.13%, 7.06%, and 3.06% when BCMR was activated. As to the reason for the extended lifespan shown in the experimental values, it is thought that the lifespan can be extended by lowering the selection frequency of the corresponding blocks between the hot and cold blocks through selecting hot blocks in garbage collection and cold blocks in wear leveling. In addition, because BCMR takes advantage of the existing garbage collection and wear leveling techniques, we were able to confirm further performance improvements through experimental results.

Fig. 4 shows the block standard deviation, which was reduced by up to 6.52%, 4.16%, 11.44%, and 7.51% when BCMR was implemented. This reduction of the block standard deviation can be considered as a meaningful indicator because it directly affects the lifespan. Although reducing the block standard deviation can extend the NAND flash memory life, there is a risk of NAND flash memory page migration overhead increasing during processing. Therefore, the algorithm that is used must consider this risk.

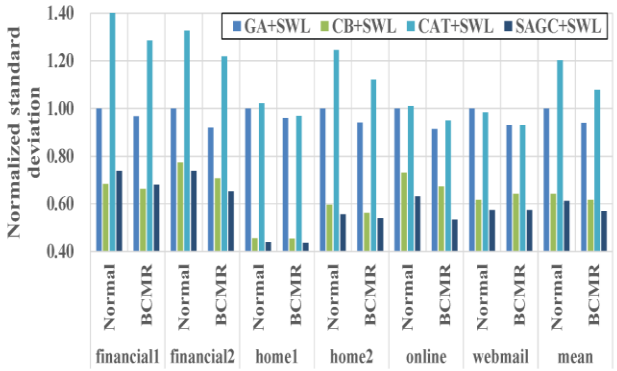

Fig. 4. The normalized standard deviation based on GA and SWL

Fig. 5 shows the average number of page migrations during garbage collection and wear leveling, which was reduced by up to 0.01%, 0.77%, 1.65%, and 0.33% when BCMR was activated.

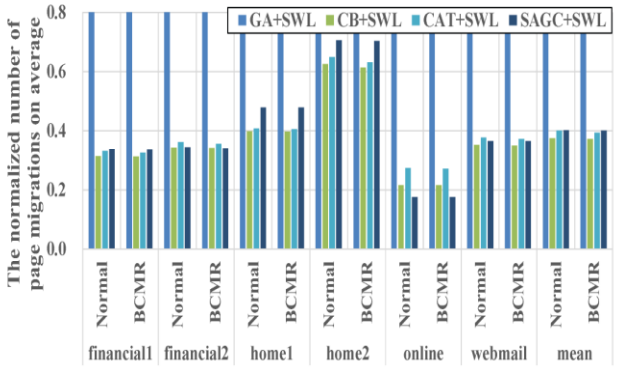

Fig. 5. The normalized number of page migrations on average

The overhead required to perform BCMR proposed in this paper is the use of separate memory. The amount of memory used to operate BCMR depends on the number of pages of the NAND flash memory used. Memory usage was measured based on the NAND flash memory used in the experimental environment; about 512 bytes (4 bytes × 128 pages) were used. Considering the total amount of memory used in the computer, this can be judged to have almost no effect on the operation of the system because of its small size. Accordingly, the method proposed in this paper is proven to extend the life of NAND flash memory.

5. Conclusions

This paper proposes the BCMR method to prevent interference and ensure isolation in garbage collection and wear leveling. The proposed BCMR monitors changes in the block erase count during garbage collection and uses this information to select hot blocks by constraining the selection of sacrifice blocks. The performance evaluation results show that the BCMR method extended NAND flash memory lifespan. The average lifespan of NAND flash memory was extended by 7% when BCMR was activated. This paper shows the block standard deviation, which was reduced by up to 11% when BCMR was implemented. This reduction of the block standard deviation can be considered as a meaningful indicator because it directly affects the lifespan. And It shows the average number of page migrations during garbage collection and wear leveling, which was reduced by about 1% when BCMR was activated. Future studies will be required to ascertain in a new hybrid buffer policy in the SSD with server computing system. An SSD with hybrid buffer memories is actively researching to reduce the overall latency in server computing systems. However, existing hybrid buffer policies caused many swapping operations in pages because it did not consider the overall latency such as read/write operations of flash chips in the SSD. Future

studies the clock with hybrid buffer memories for a new hybrid buffer policy in the SSD with server computing systems.

6. Acknowledgment

This research was supported by the Yeungnam University College Research Grants in 2019.

References

1. Myungsub Lee, "A Monitoring Methodology based on Block Erase Count for Classifying Target Blocks", Smart Convergence of Culture Technology Letters, SCCTL06, pp.265-270, 2020.
2. Myungsub Lee, "Sampling-Based Block Erase Table Method in Wear Leveling Technique for Flash Memory", Journal of Engineering and Applied Sciences, 13(4), pp. 1023-1028, 2018.
3. Myungsub Lee, "A Wear Leveling Technique based on SBET for Flash Memory", Advanced and Applied Convergence Letters, AACL10, pp. 113-115, 2017.
4. Myungsub Lee, "A Tracking Cold Blocks for Static Wear Leveling in FTL-based NAND Flash Memory", The 6th International Symposium on Advanced and Applied Convergence, ISAAC2018, pp. 50-52, 2018.
5. Ma. D, Feng. J, & Li. G, "A survey of address translation technologies for flash memories", ACM Computing Surveys (CSUR), Vol. 46, No. 3, 2014.
6. Chang. B, Wang. Z, Chen. B, & Zhang. F, "Mobipluto: File system friendly deniable storage for mobile devices", Proceedings of the 31st Annual Computer Security Applications Conference, pp. 381-390, 2015. (4)
7. Athanassoulis. M, & Ailamaki. A, "BF-tree: approximate tree indexing", Proceedings of the VLDB Endowment, Vol. 7, No. 14, pp. 1881-1892, 2014.
8. Chung. T. S, Park. D. J, Park. S, Lee. D. H, Lee. S. W, & Song. H. J, "A survey of flash translation layer", Journal of Systems Architecture, Vol. 55, No. 5, pp. 332-343, 2009.
9. Hachiya. S, Johguchi. K., Miyaji. K, & Takeuchi. K, "TLC/MLC NAND flash mix-and-match design with exchangeable storage array", 2013 International Conference on Solid State Devices and Mater, pp. 894-895, 2013.
10. Yang. M. C, Chang. Y. M, Tsao. C. W, Huang. P. C, Chang. Y. H, & Kuo. T. W, "Garbage collection and wear leveling for flash memory: past and future", SACM Computing Surveys (CSUR), pp. 66-73, 2014.
11. Kwon. O & Koh. K, "Swap space management technique for portable consumer electronics with NAND flash memory", IEEE Transactions on Consumer Electronics, Vol. 56, No. 3, 2010.
12. Chang. Y. H, Hsieh. J. W & Kuo. T. W, "Improving flash wear-leveling by proactively moving static data", IEEE Transactions on Computers, Vol. 59, No. 1, pp. 53-65, 2010.