Dual Address Mapping Method for Efficient Wear-Leveling of SSDs

Jiwon Lee, Hongchan Roh, and Sanghyun Park

Abstract— **Flash memory based Solid-State Disks(SSDs) are currently being considered as a promising candidate for replacing hard disks, due to several superior features such as significantly shorter access time, lower power consumption and better shock resistance. SSDs, however, have some drawbacks as well, including the slow write time for random-writes and limited erase-count of a block. In order to improve write throughput, SSDs take advantage of inter-leaving method, which writes data on several NAND-flash chips in parallel. For interleaving method being applied, the granularity of write-operations is increased to as many blocks as up to the number of flash memory chips constituting a SSD, and therefore the logical block size is increased according to the increased granularity. Increase of write-operation granularity results in some problems. Even small-size write-operations cause many blocks to be erased at the increased granularity. Consequently, the limited erase-count of a block is consumed fast. Specifically, a certain block consisting of a logical block can be worn out faster than the other blocks of the logical block, when wear-leveling techniques are performed on logical blocks. In order to address these problems caused by the large logical block size, we propose a novel method named dual address mapping that perform a write operation at the unit of a block as well as a logical block and manages not only erase counts of logical blocks but also those of blocks.**

I. Introduction

Flash memory is a non-volatile semiconductor with shock resistance, low power consumption and small sizes [9]. Reflecting these superior characteristics, recent mobile devices such as mobile phones, PDAs, MP3 adopted flash memory as their storage media. Moreover the advent of flash based-SSDs (Solid-State Disks) is hastening to replace hard disks with SSDs, not only in personal computers but even in enterprise servers [10]. Besides these characteristics, SSD has faster data processing speed with no mechanical access time than hard disks. Due to these superior characteristics and the trend that SSD is becoming to have bigger capacity and cheaper price, it is expected that SSDs will replace the hard disks in the near future [7].

Table 1 shows terminologies used in this paper. A flash memory chip is constituted with many blocks, and one block is constituted with several pages [10]. Page is the basic unit of read and write operations. Figure 1 shows this structure of flash memory. About 128bytes of spare area per page exists on flash memory in order to store a sort of meta-data of data pages [3].

TABLE 1
TERMINOLOGIES USED IN THIS PAPER

Notation	Description
Page	granularity of a read and write operation
Block	granularity of a erase operation
Logical block	concept to manage several blocks together
Free block	write-possible block

Flash memory has a peculiarity which is the difference between the granularities of a read, write, and erase operation. Because flash memory cannot execute in-place update, the difference of granularity between read, write, and erase operation is one of drawbacks of flash memory [6]. To perform an update operation, existing data in block should be eliminated before writing a new data or a new data should be stored in another free block. However, because of erase count of each block being limited from ten thousands to one million [5], the difference of the erase count between each block should be considered when storing data. That is, a technique that divides data evenly among all blocks should be applied and this technique is called wear-leveling. Consumed erase count of each block is stored in spare area, and wear-leveling technique exploits this information. This spare area also stores ECC (Error Correcting Code) for preventing occurable errors when data is read [3].

In flash memory, write latency for 2KB page, about 200µs, are much slower than read latency which is about 25µs [2]. To complement these drawbacks, the SSD has been developed as a permanent storage media that provide higher write throughput and larger storage space, by chunking several flash chips and using new mapping technologies. As shown in figure 2, the SSD controller receives data from the host interface, and then data is transferred to flash memory

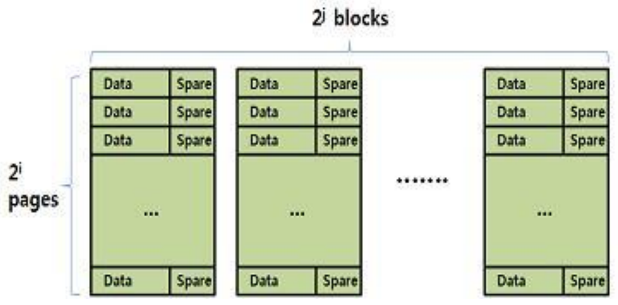

Fig. 1. Structure of NAND-flash memory chip

J. Lee is with the Computer Science Department, Yonsei University, Seoul, Korea (corresponding author to provide phone: 82-2-2123-7757, fax: 82-2-365-2579, e-mail: sppong@cs.yonsei.ac.kr).

H. Roh is with the Computer Science Department, Yonsei University, Seoul, Korea (e-mail: fallsmal@cs.yonsei.ac.kr).

S. Park is with the Computer Science Department, Yonsei University, Seoul, Korea (e-mail: sanghyun@cs.yonsei.ac.kr).

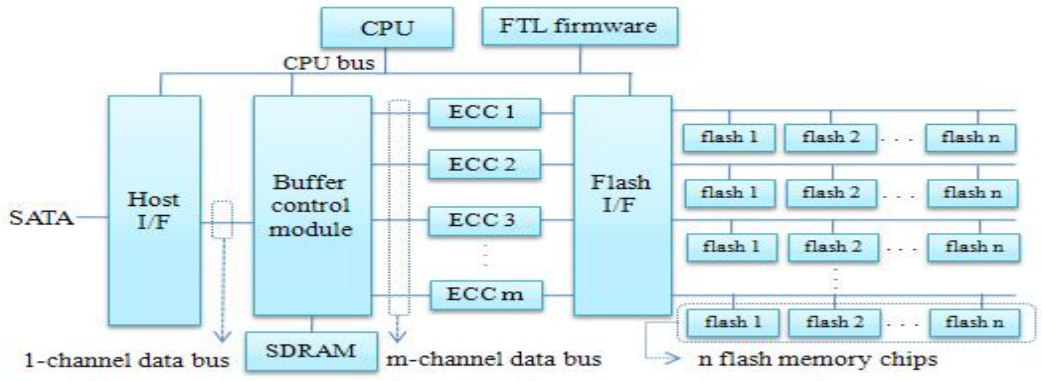

Fig. 2. General structure of SSD

interface through the data bus channel linked to host interface. Writing performance can be enhanced by using inter-leaving method, which makes it possible for the SSD controller to simultaneously write data on the flash memory chips in parallel [2]. Several blocks have to be joined together in one logical block so that the inter-leaving method efficiently works. However, the large size of a logical block causes several problems. Update operation smaller than the size of logical block makes other data that has not been updated copied. These unnecessary write operations accompanied by later erase operations can make write performance slower and erase-count consumed faster [2]. Moreover, these also make the difference between each block's erase-count in the same logical block. Therefore, if a certain block is worn out significantly faster than the others, then the entire logical block containing the prematurely aged block will not be used as the storage area by the SSD controller, wasting the other blocks.

In this paper, we propose a new method which can cope with these problems caused by logical block being larger. Our method provides concrete techniques that perform a write operation at the granularity of a block as well as a logical block, and maintain not only the erase-count of each logical block but the erase count of each block consisting of the logical block.

The rest of the paper is organized as follows. Related work is presented in Section 2. Section 3 describes the proposed method of address mapping. We estimate performance of dual address mapping method in section 4.We conclude in Section 5.

II. Related Work

Sandisk, a company producing flash memory chips and SSDs, made an application for a patent that includes wear-leveling technique. This maintains several blocks as one logical block named a bank [4]. In this patent, averages of erase counts of blocks that each bank contains are stored in each bank. A method that made the erase counts of blocks not be widen was proposed. This paper named the bank which has largest erase count 'max bank', and the smallest 'min bank'. When a certain condition is satisfied, a max bank is switched to a spare bank which is a saved bank in order to prevent a gap between erase counts from dilating. Before switching a max bank with a spare bank, data of max bank are copied to spare bank, and the max bank is erased for making the space free to write. This process makes the spare bank have the largest erase count of all the banks. If erase count of a max bank is greater than a spare bank by a certain threshold value, the max bank is switched with the spare bank. However, this patent has a problem that any block of the bank can be worn out fast, since erase count of blocks belong to the same bank isn't considered,.

MTRON, another company producing SSD, proposed the HYDRA architecture. This architecture can execute write operations fast by using inter-leaving method and using buffering method [2]. Buffering method is adopted so that small-size write operations are collected in the buffer and later the gathered data can be flushed at the same time. Buffering method complements the weak point of SSDs caused by small-size write operations. Small and random writes, however, still degrade write performance, and data in buffer can be lost by system crashes.

III. Dual Address Mapping Method

To address aforementioned shortcomings of SSDs, we suggest dual address mapping method.

The proposed method has two major purposes. First, we made an effort to deal with drawbacks regarding small-size write by supporting both logical block-based write operation and block-based write operation. Larger logical blocks accompanied with inter-leaving method enhances write throughput in case of sequential writes, but no write performance enhancement can be expected when small-size and random write operations dominates the workload pattern. When small-size and random write operations occurred, all the data including not only updated data by the write operations but also other data that has not been updated have to be copied to a new logical block. Therefore, we designed

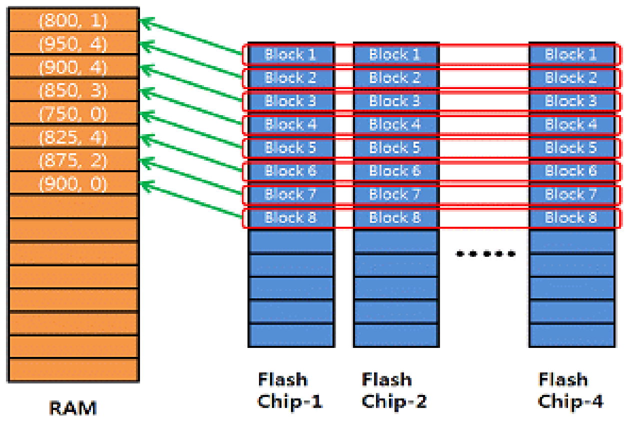

Fig. 3. Process of initializing system

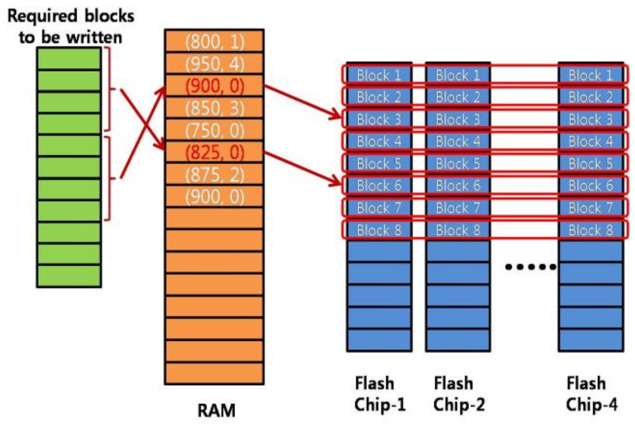

Fig. 4. Process of logical block-based writing operation

dual address mapping method to enable for block based writing not to copy all the data under small-size write operations. Second, in order to prevent erase-count of blocks not be widen, dual address mapping method applies wear-leveling technique on each block. Most previous studies haven't considered the erase count of each block consisting of a logical block but the erase count of the logical block. However, the proposed method prevents any block of logical block worn out earlier than the other blocks by considering the erase count of each block that belongs to the logical block.

In dual address mapping method, an average of erase counts of the blocks consisting of each logical block is calculated and loaded into RAM of SSD controller when system is initialized. Next, the number of free blocks in each logical block which can be written is loaded into in RAM. Figure 3 illustrates the process in which the information of all the logical blocks is stored in RAM in the SSD controller consisting of four flash memory chips. The form of the information is represented as (an erase count, the number of free blocks). Writing process is divided into two steps, logical block-based and block-based writing. The process of writing is presented in Algorithm 1.

If the data size for a writing operation is larger than or is the

Algorithm 1.

Process of writing operation

```
1: Calculate the number of required logical block and the
   number of required block denoted as $M$ and $N$
2: Choose $M$ logical blocks satisfying two conditions
3: while ( $N <= 0$ )
4:    Choose target logical block that has free blocks and its
      erase count is the smallest among all the logical blocks
5:    Set $T$ as the number of free blocks belong to the target
      logical blocks
6:    if( $T > N$ )
7:       Load erase count of free blocks into RAM
8:       Perform block-based write processes in increasing
         order of the free blocks' erase counts $N$ times
9:    else
10:      Perform block-based write processes $T$ times
11:  $N = N - T$
```

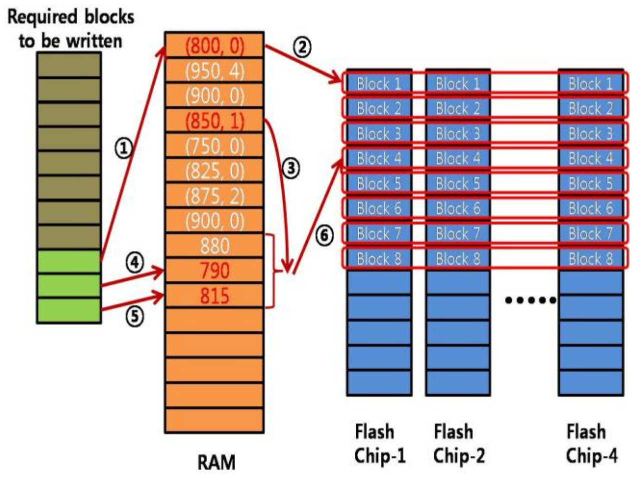

Fig. 5. Process of block-based writing operation

same as the logical block size, logical block-based writing process is performed. The number of required logical blocks for a write operation, N_L, can be represented as equation (1), where S_W, S_L, and S_B denotes the data size for a write operation, logical block size, and block size, respectively.

$$N_L = \lfloor S_W / S_L \rfloor \quad (1)$$

After performing logical block-based writing, remaining data also have to be written. The size of remaining data for the write operation is calculated as the following equation.

$$S_W - (\lfloor S_W / S_L \rfloor \times S_L) \quad (2)$$

Through the equation (2), it can be confirmed that the size of remaining data is always smaller than S_L. This remaining data is written by block-based writing process to address small-size write problems. N_B denotes the number of required blocks for the write operation. N_B can be calculated as equation (3).

$$NB = \lceil (S_W - (\lfloor S_W / S_L \rfloor \times S_L)) / SB \rceil \quad (3)$$

The write operations are performed, according to the number of required logical blocks, N_L, and that of required blocks, N_B, which are derived by equation (1) and (3). Logical blocks and blocks to be written are located by the information of logical blocks stored in RAM. The logical blocks required for write operations are computed satisfying two conditions: First, every block included in a logical block should be a free block. Second, the logical block which has the smallest erase count has the highest priority to be written. These are represented as step 1 and 2 in Algorithm 1. Figure 4 demonstrates the process of logical block-based writing when eleven blocks are required to be written on SSDs. Since one logical block contains four blocks, logical block-based writing is performed on eight blocks of required blocks. Following the aforementioned conditions, two logical blocks each of which has four free blocks along with higher priority than others are written.

To handle the remaining data after performing logical block-based writing, block-based writing process is executed. Before carrying the process, a target logical block should be chosen first. The target logical block has to satisfy two constraints: First, it should have enough free blocks. Second its erase count has to be the smallest among all the logical blocks in the aspect of wear-leveling. When the number of free blocks that belong to the target logical block is less than N_B, all the free blocks of the target logical block are written. The above process is performed until the number of free blocks that belong to the target logical block is larger than that of the remaining blocks. On the other hand, when the number of remaining blocks is less than that of the free blocks that belong to the target logical block, the free blocks of the target logical block are written in increasing order of the free blocks' erase counts. The step 3 to 11 of Algorithm 1 explains block-based writing process. Figure 5 shows the steps of block-based writing process for the remaining data after the logical block-based writing process is performed. The three remaining blocks are also written on the target logical block which has free blocks and the smaller erase count than any other logical blocks. As shown in ① of Figure 5, only one free block exists on the logical block having the smallest erase count whereas three blocks to be written are left. Therefore, write operations are performed as shown in ② of Figure 5. Since two blocks still remain, another target logical block has to be chosen. The new target logical block has three free blocks, so the two of them, which are the blocks having the first and second smallest erase count, are written. The step ③ of Figure 5 demonstrates the process of reading free blocks' information in order to choose two blocks to be written. The step ④, ⑤, and ⑥ show the process in which the chosen two blocks are written.

Dual address mapping method can complement the drawbacks of SSDs regarding small-random writes by providing block-based writing process as well as logical block-base writing process. Besides, this can address the problem that a certain block is worn out faster than others, as well.

IV. Performance Evaluation

For there is no mapping algorithm for SSD publically available, it is difficult to evaluate the performance by comparing the proposed method with other SSD mapping algorithms. Also, adjusting internal logic of SSD is difficult. Because of these reasons, we chose the evaluation method that estimates the performance of dual address mapping method by theoretical analysis. Logical block-based mapping algorithm, block-based mapping algorithm and log-structured mapping algorithm of flash memory are compared in the analysis. Table 2 shows the performance comparison between dual address mapping and other mapping algorithms.

If N blocks are required to be written, N can be represented as (4).

$$N = kA + B \tag{4}$$

In formula (4), kA represents the number of required blocks to perform logical block-based writes and B represents that to perform block-based writes, where k is the number of flash memory chips consisting of SSD. Besides them, A denotes the number of logical block-based writes and B is always smaller than k.

Compared with the mapping algorithm which manages only the unit of logical block, the proposed method has the outstanding merit in the aspect of small-random write. $k(A+1)$, the number of needed block on logical block-based sequential write, is greater than $kA+B$, the number of needed block on sequential write of the proposed method. On the other hand, $A+1$, which denotes the number of memory accesses by logical block-based sequential write, is less than $A+B$, which denotes the number of memory accesses by the proposed method, in worst case. Consequently, the sequential write performance of logical block-based mapping algorithms and the proposed method are approximately the same. In the view point of small-random writes, however, the difference of performance is much clear. For enhancement of write performance to small-random writes, the buffer size should be expanded. As the proposed method supports block-based writing, the buffer size can be smaller than that of logical block-based mapping algorithm, and unnecessary writes and erase operations accompanied by small size updates can be reduced to up to $1/k$.

Compared with the mapping algorithms which manage

TABLE 2
Performance comparison with other algorithms

mapping algorithm / Attribute	Dual Address mapping method	Logical block-based mapping algorithm	Block-based mapping algorithm	Log-structured mapping algorithm
Usage of RAM (# of information should be loaded)	N/k	N/k	N	N/k
# of memory accesses on sequential write	A+B (in worst case)	A+1	kA+B	A+B
# of needed blocks on sequential write	kA+B	k(A+1)	kA+B	kA+B
# of unnecessary copies on small-random write in worst case	One block	k blocks	One block	None (but this accompanies merge operations)
# of blocks should be read to acquire data of M blocks, where M=cX+Y(<c)	cX+Y	c(X+1)	cX+Y	c(X+1) + w (w : # of log blocks have to be written together)

only the unit of block, the proposed method has advantages in some aspects. First of all, the main memory usage to store logical-to-physical mappings can be considerably reduced as far as *1/k*, since meta data of all the blocks on block-based mapping algorithm should be loaded into the main memory. For sequential reads and writes, dual address mapping method also has better performance. This requires $A+B$ memory accesses, whereas block-based mapping algorithm requires $kA+B$ memory accesses. The performance of small-random writes on these two algorithms is almost the same.

Dual address mapping algorithm can be compared with log-structured mapping algorithm applied in flash memory such as BAST [8], FAST [6]. Since BAST and FAST are designed for flash memory, these have to be adapted to the SSD architecture. Log-structured mapping algorithm includes log blocks in order to store logs. Therefore, the entire capacity of SSDs where data can be stored is decreased as much as the total size of log blocks. Since log-structured mapping algorithms stores updated data into log blocks by page unit, given small-random write operations, these methods exhibit seemingly better performance than dual address mapping, however. On the other hand, the adoption of the log blocks accompanies merge operations that select useful data from data block and logical block, and then copy them into another data block so that they can recycle them as free blocks for future write operations. This leads performance degradation of log-structured mapping algorithms. Likewise, read performance of these methods is expected to be lower than dual address mapping due to the fact that one data block and log block have to be read together per a read request.

V. CONCLUSION

Flash memory based Solid-State Disks(SSDs) are expected to gradually replace hard-disks. Flash memory has fast data processing speed without mechanical access time, low power consumption, and shock-resistance. To improve write performance, SSDs employ inter-leaving method which can simultaneously write several data in parallel. However, inter-leaving method makes logical block larger. This larger logical block causes write performance degradation by non-necessary copy operations. SSDs have another problem that a certain block that belongs to a logical block can be worn out fast. To address these problems of SSDs, we propose the dual address mapping method, which can manage not only logical blocks but blocks. The proposed method enhances write performance by providing block-based writing as well as logical block based writing. This also alleviates the problem that a certain block that belongs to a logical block is worn out earlier than others, by considering the erase count of a block as well as that of a logical block.

As future work, we are planning to design a more efficient system by considering more various performance factors of SSDs such as RAM size and garbage collection operations.

REFERENCES

[1] Hongseok Kim et al., "Development Platforms for Flash Memory Solid State Disks", IEEE ISORC, pp. 527~528, 2008

[2] Young-Hyun Bae, “Design of A High Performance Flash Memory-based Solid State Disk”, KIISE, Vol 25, No. 6, pp. 18~28, 2007

[3] Nitin Agrawal et al., "Design Tradeoffs for SSD Performance", USENIX, pp. 57~70, 2008

[4] Karl M. J. Lofgren, "Wear Leveling Techniques for Flash EEPROM Systems", United State Patent, No. 6,230,223, 2001

[5] Eran Gal et al., "Algorithms and Data Structures for Flash Memories", CSUR, Vol 37, No. 2, pp. 138~163, 2005

[6] Sang-Won Lee et al., "A Log Buffer-Based Flash Translation layer Using Fully-Associative Sector Translation", ACM Transactions on Embedded Computing Systems, Vol 6, No. 3, Article 18, 2007

[7] Goetz Graefe, Hewlett-Packard Laboratories, "The Five-Minute Rule 20 Years Later: and How Flash Memory Changes the Rules", ACM QUEUE, Vol 6, Issue 4, pp. 40~52, 2008

[8] Jesung Kim et al., “A Space-Efficient Translation Layer for Compactflash Systems”, IEEE Transactions on Consumer Electronics, Vol 48, No. 2, 2002

[9] Jeong-Uk Kang et al., “A Superblock-based Flash Translation Layer for NAND Flash Memory”, Proceedings of the 6th ACM & IEEE International conference on Embedded software, pp. 161~170, 2006

[10] Aayush Gupta et al., “DFTL : A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings”, Proceeding of the 14th international conference on Architectural support for programming languages and operating systems, pp. 229~240, 2009