International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 22 (2015) pp 43066-43068


# Write Pattern based Hot/Cold Separation for Flash Translation Layer

**S. Hwang,**
*Undergraduate, Department of Electronic Engineering, Seoul National University of Science and Technology*
*Seoul, South Korea*

**T. Jung[1],**
*Undergraduate Department of Electronic Engineering, Seoul National University of Science and Technology*
*Seoul, South Korea*

**I. Shin[2]**
*Associate Professor, Department of Electronic Engineering, Seoul National University of Science and Technology*
*Seoul, South Korea ilhoon.shin@snut.ac.kr*

**Abstract**
Solid State Drives(SSDs), which are composed of NAND flash memory, embed a firmware called flash translation layer(FTL) to support the standard block device feature. Because FTL performs an out-of-place update, which writes new data to clean pages, a garbage collection is performed whenever clean pages are lacking. The garbage collection is a main performance overhead of SSDs. In order to reduce its overhead, we should increase the invalidation ratio of victim blocks. For this purpose, we propose a scheme that causes a skewed invalidation ratio over all the blocks by separating cold data from hot data with epoch. The performance evaluation result on the simulator shows that the proposed scheme improves the overall performance of SSDs up to 30.5%.

**Keywords:** Epoch, Hot/cold, page mapping, flash translation layer, NAND flash memory.

## Introduction

SSDs are substituting HDDs because of fast speed, low energy consumption, and shock resistance. SSDs are composed of NAND flash memory which has different characteristics from standard block devices. Therefore, in order to use them as standard block devices, we need an intermediate firmware called FTL between file system and NAND flash memory. FTL emulates standard block device interface, especially an overwrite feature.

NAND flash memory is composed of pages and blocks, as shown in Figure 1. Read/write unit is a page, and a block is an erase unit. Data should be sequentially written within a block. In NAND flash memory, the overwrite is not supported. In other words, pages should be first erased to be re-written. Therefore, on a write request, FTL performs an out-of-place update, which writes new data to a new clean page and invalidates the obsolete data.
By the continuous out-of-place update, invalid pages are scattered over an entire NAND flash memory, and clean pages are eventually lacking. This triggers a garbage collection which reclaims invalidated pages to clean pages. It first selects a victim block and then moves valid pages to a clean extra block. Finally, pages of the victim block are reclaimed to clean by the block erase. The garbage collection is a main performance overhead of SSDs, and it is important to reduce both the frequency and the latency of the garbage collection. In order to improve them, we have to select the most invalidated block as victim because the garbage collection time increases proportionally to the number of valid pages in the victim and because the number of reclaimed clean pages is the same with the number of invalidated pages. Therefore, separating frequently updated data (hot data) from rarely updated data (cold data) contribute to reduce the overhead of the garbage collection. In this work, we present a scheme to predict the hotness of data and to separate hot data from cold data.
The rest of the paper is organized as follows. Section 2 explains page mapping scheme which is a representative FTL scheme. Section 3 describes the presented hot/cold separation scheme. Section 4 presents a performance evaluation result on a simulator. Finally, section 5 concludes the work.

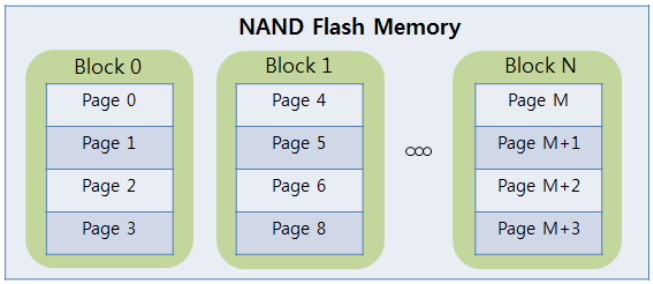


**Fig.1. NAND flash memory structure**

## Page Mapping Scheme

In the out-of-place update, the mapping information between logical sectors and their physical locations should be maintained. FTL is classified to page mapping scheme [1],

[1] He is current working for Samsung Electronics
[2] Corresponding Author

block mapping scheme [2], and hybrid mapping scheme [3, 4], according to the size of the mapping unit. Among the schemes, the page mapping scheme delivers the best performance and it is adopted in most commercial SSDs.

The page mapping scheme uses a NAND page as a mapping unit. When a request arrives, It converts a sector number to a logical page number (LPN) and attains its real address that is denoted as a physical page number (PPN) by referring to the mapping table, Then, it reads and writes NAND flash memory using the PPN. Figure 2 illustrates an example of a writing process. When file system sends a write request for LPN 7, FTL checks whether previous data of LPN 7 exist or not from the mapping table. If it exists, the previous PPN 17 is invalidated. The new data are written in a next clean page of the current write block, PPN 56 in this example, regardless of its hotness. The mapping table is updated accordingly. Thus, hot data and cold data can be mixed in a block, which restricts an efficiency of the garbage collection.

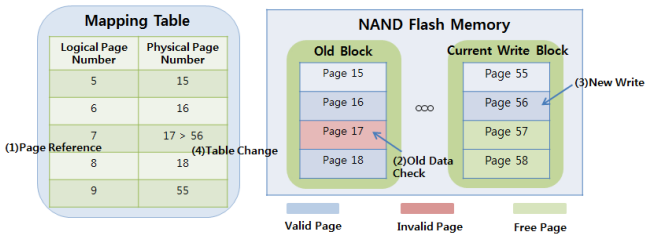


**Fig.2. NAND flash memory structure**

## Proposed System

As described in section 1, the overhead of the garbage collection is reduced by increasing the invalidation ratio of victim blocks. We can increase the invalidation ratio by collecting hot data in a block and by separating cold data.

For this purpose, hot/cold separation scheme predicts the hotness of data using a write pattern and separates cold data from the others. For data separation, it uses two working blocks: one is for hot data and the other is for cold data (figure 3). The hotness of data is evaluated based on the following simple assumptions. First, the data written for the first time are assumed to be cold because the hotness cannot be evaluated. Second, if data are modified, they are regarded as hot because they are likely to be re-written in a future. Therefore, first written data are gathered in the cold working block, while the overwritten data are gathered in the hot working block.

If the data are modified after very long time, it is hard to regard as hot data. Therefore, we employ the epoch to separate data more accurately. If the data are modified after the epoch, they are regarded as cold data. In short, hot data are the data modified within the epoch. Cold data are the data written first time or modified after the epoch. Therefore, the invalidation ratio of the hot blocks will be greater than the cold blocks, and the hot blocks will be selected as victim at the garbage collection.

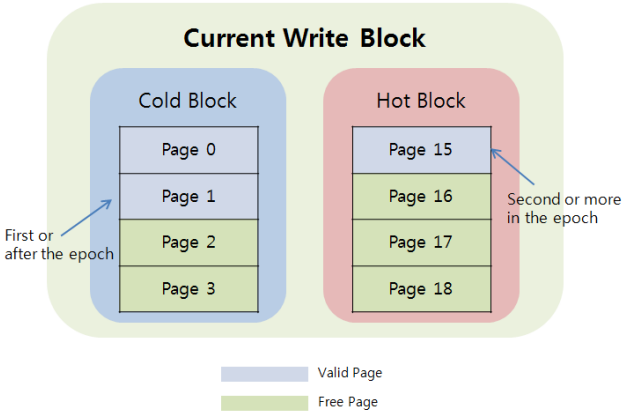


**Fig.3. NAND flash memory structure**

## Simulation

We use four workloads to evaluate the performance. These workloads were attained with diskmon tool in window PCs. Table 1 shows the detailed description about the workloads. WinXP_55GB is a workload file which collects I/O pattern in 55 GB partition where windows XP is installed. Likewise, winXP_59GB contains IO pattern when installing windows XP in 59GB storage partition. win7_45GB and win7_59GB collect IO pattern when installing windows 7 in 45GB and 59GB storage partition respectively. A To measure the performance, we implemented the page mapping scheme and our hot/cold separation scheme in simulator and measured the total read, write and erase count while inputting the workloads. Then total elapsed time is calculated based on the latencies of NAND flash memory in Table 2 [5].

Figure 4 shows write throughput calculated by (2). The write throughput is improved 43.8% in winXP_55GB workload, 10.2%, 0.6% and 1.3% in each workload. Figure 5 shows the average ratio of invalid page in victim block which is the main factor of improving the performance. The ratio is increased up to 47.7% in winXP_55GB workload. The frequency of garbage collection is also reduced as shown in Figure 6 and it is reduced up to 32.8% in winXP_55GB. By increasing the average invalid page ratio and reducing garbage collection frequency, we can reduce the overhead of garbage collection and achieve the improvement.

l the target partitions were formatted with NTFS file system.

The result shows that our hot/cold separation scheme is very effective in window XP workloads. However, the improvement is not so significant in window 7 workloads because they do not have a lot of cold data. The improvement drawn by the hot/cold separation is related the ratio of the cold data in each workload. In window XP workloads, the ratio of the cold data were high, and thus the benefit of separating the cold data from the hot data was also significant.

**TABLE.1. Workload description**

| | winXP_55GB | winXP_59GB | win7_45GB | win7_59GB |
|---|---|---|---|---|
| Total read bytes | 147.56GB | 113.87GB | 281.59GB | 124.33GB |
| Total written bytes | 153.95GB | 119.66GB | 371.25GB | 105.35GB |

**TABLE.2. Latency of NAND operation**

| Operation | Latency |
|---|---|
| Read ($T_r$) | 25us |
| Write ($T_w$) | 220us |
| Erase ($T_e$) | 1.5ms |
| Transfer ($T_t$) | 121.6us |

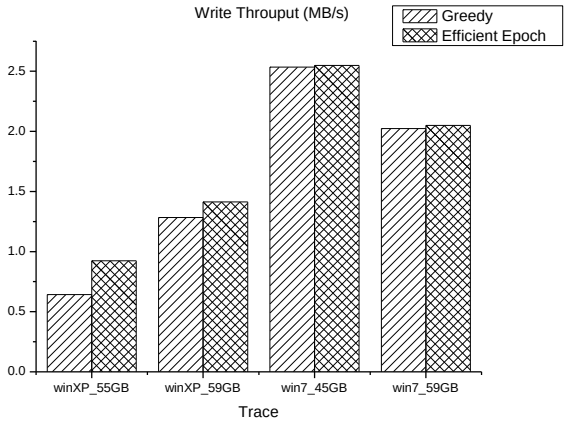


**Fig.4. Write throughput comparison**

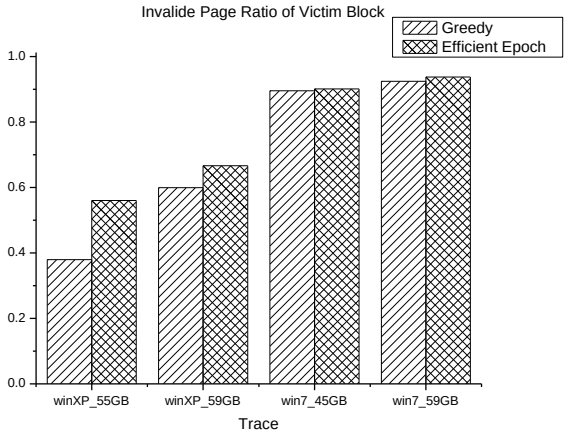


**Fig.5. Comparison of average invalid page ratio of victim block**

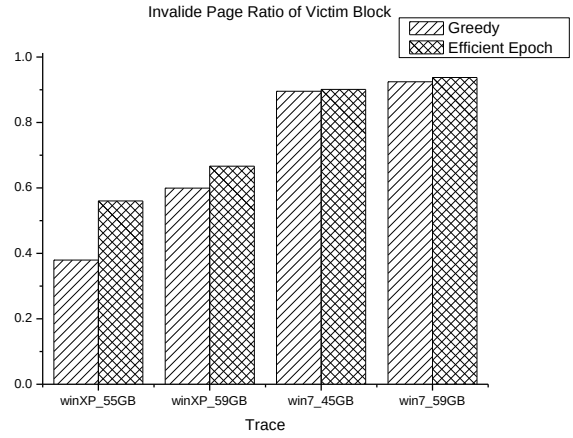


**Fig.6. Garbage collection count comparison**

## Conclusion

The garbage collection time, which is the main performance overhead of the page mapping scheme, could be reduced by increasing the invalidation ratio of victim blocks. In this work, we proposed the scheme that separated cold data from hot data and thereby made the invalidation ratio of blocks skewed. The skewed invalidation ratio increased the invalidation ratio of victim blocks. The performance evaluation result on the simulator shown that the proposed scheme improved the performance of SSDs significantly in certain workloads. However, its gain was restricted in the workload such as win7_45GB, where the ratio of cold data was low.

## Acknowledgements

This study was supported by the Research Program funded by the Seoul National University of Science and Technology.

## References

[1] A. Ban, “Flash file system”, Unites States Patent, no. 5,404,485, 1995.

[2] A. Ban, “Flash file system optimized for page-mode flash technologies”, United States Patent, no. 5,937,425, 1999.

[3] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho, “A space-efficient flash translation layer for compactflash systems”, IEEE Trans. Consumer Electron., vol. 48, no. 2, pp. 366–375, 2002.

[4] S. W. Lee, D. J. Park, T. S. Chung, W. K. Choi, D. H. Lee, S. W. Park, and H. J. Song, “A log buffer based flash translation layer using fully associative sector translation”, ACM Trans. Embedded Computing Systems, vol. 6, no. 3, Jul. 2007.

[5] Micron, <MT29F4G08AAA>.