ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.13, No.1, 1–8

Clustering data according to update frequency to reduce garbage-collection overhead in solid-state drives

Jaehyun Kim and Ilhoon Shin[a)]

Department of Electronic Engineering, Seoul National University of Science & Technology, Gongleung-Dong, Nowon-Gu, Seoul, 139–744, South Korea

a) *ilhoon.shin@snut.ac.kr*

Abstract: Garbage collection, which entails multiple page copies and a block erase, is a major source of performance fluctuation and degradation for NAND flash memory-based solid-state drives. This work aims to reduce its overhead by generating a skewed distribution of valid pages over all the blocks. Therefore, we propose classifying data as hot, warm, or cold according to their update frequencies, and to cluster them into different blocks. Our performance evaluation shows that the proposed scheme reduces the total garbage-collection count up to 43.4%, compared to the original page-mapping scheme, for an average performance improvement of up to 34.5% without any additional memory overhead.

Keywords: flash translation layer, NAND flash memory, garbage collection, hot/cold, update frequency

Classification: Storage technology

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse and R. Panigrahy: Proc. of USENIX Annual Technical Conference (2008) 57.

[2] A. Kawaguchi, S. Nishioka and H. Motoda: Proc. of USENIX Annual Technical Conference (1995) 155.

[3] A. Ban: United States Patent, No. 5,404,485 (1995).

[4] A. Ban: United States Patent, No. 5,937,425 (1999).

[5] J. Kim, J. M. Kim, S. Noh, S. L. Min and Y. Cho: IEEE Trans. Consum. Electron. **48** (2002) 366. DOI:10.1109/TCE.2002.1010143

[6] S. W. Lee, D. J. Park, T. S. Chung, D. H. Lee, S. W. Park and H. J. Song: ACM Trans. Embed. Comput. Syst. **6** (2007) 18. DOI:10.1145/1275986.1275990

[7] I. Shin and Y. Shin: IEICE Electron. Express **11** (2014) 20130942. DOI: 10.1587/elex.10.20130942

[8] I. Shin: IEICE Trans. Inf. & Syst. **E97-D** (2014) 2844. DOI:10.1587/transinf. 2014EDP7075

[9] I. Shin: IEEE Trans. Consum. Electron. **57** (2011) 1728. DOI:10.1109/TCE. 2011.6131147

[10] T. Jung, Y. Lee, J. Woo and I. Shin: Lecture Notes in Electrical Engineering **279** (2014) 141. DOI:10.1007/978-3-642-41674-3_21

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

[11] The OpenSSD Project: http://www.openssd-project.org/wiki/The_OpenSSD_Project.

1 Introduction

Solid-state drives (SSDs) that use NAND flash memory as storage media have advantages, such as high I/O operations per second (IOPS) and low energy consumption, compared to hard-disk drives (HDDs). Thus, they are replacing HDDs in various computing environments, including transaction-processing systems and laptop computers [1].

Although NAND flash memory is similar to an HDD in that both of them are non-volatile media, it has several unique characteristics. First, it is an electrically programmable/erasable read-only memory that does not support an overwrite operation. Once a cell is written, it cannot be re-written without first being erased. Second, an erase unit is larger than a read/write unit. NAND flash memory consists of multiple blocks that are an erase unit; a block consists of multiple pages that are a read/write unit. Due to these unique features, traditional file systems that are designed for HDDs do not work correctly on NAND flash memory. Thus, SSDs employ intermediate firmware called the flash-translation layer (FTL) inside the devices. The main role of the FTL is to export NAND flash memory like a standard block device that consists of multiple sectors and supports the overwrite operation.

The FTL emulates the overwrite operation with an out-of-place update. On a write request, it searches for clean pages and writes data to them. Thus, the location of valid data changes on every write request; the FTL tracks the current location of each logical address with a mapping table. According to the mapping granularity between a logical address and its physical address, the FTL is classified as page mapping [1, 2, 3], block mapping [4], and hybrid mapping [5, 6, 7, 8]. The page-mapping FTL uses NAND pages as a mapping unit and delivers better performance than the other schemes. The weakness is large memory consumption owing to the small mapping unit. Thus, high-end SSDs that have a large amount of memory and require high performance generally employ the page-mapping FTL [1]. This work also focuses on the page-mapping scheme.

In the page-mapping FTL, write requests are served by finding clean pages and writing data to the found pages. Thus, the response time is quite short, as long as sufficient clean pages exist. However, if clean pages are lacking, we need to perform a garbage collection to reclaim the clean pages. First, we select a victim block and copy the valid pages from the victim block to an extra clean block. Then, the victim block is erased and changed to a new extra block. The previous extra block becomes a new working block to serve write requests. This garbage collection is a major source of performance fluctuation and degradation because it entails a costly block erase and multiple page copies. Reducing its overhead is critical to improving the overall performance of SSDs.

To reduce the garbage-collection latency, we use the greedy scheme [2], which selects the most invalidated block as the victim. For example, if the victim block is fully invalidated, the garbage-collection latency is the same as the block erase

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

latency, because copying valid pages is unnecessary. Choosing the most invalidated block is also effective for reducing the garbage-collection frequency, because the number of reclaimed clean pages is inversely proportional to the number of valid pages in the victim block. However, if the valid pages are evenly distributed among the blocks, the advantage of the greedy scheme is restricted.

To maximize the effect of the greedy scheme and reduce the garbage-collection overhead, the proposed scheme uses multiple working blocks to cluster hot data, warm data, and cold data. The remainder of our paper proceeds as follows. Section 2 discusses previous work. Section 3 presents our proposed algorithm and Section 4 evaluates its performance. Section 5 presents our conclusions.

2 Related work

Previous studies to skew the invalidation ratio have tried to separate rarely updated data from the others [9, 10]. The hot/cold clustering scheme regards the valid data of the victim block as cold data, because they have not been updated for a long time [9]. To separate these cold data from the others, two working blocks—a hot block and a cold block—are used. The hot block accommodates data from write requests, and the valid data of the victim block are moved to the cold block, instead of the extra block during the garbage collection. Thus, the cold data are separated from the others. This method delivers better performance than the original page-mapping scheme. However, it is limited in that the write request data are mixed in the hot block regardless of their update frequency, and the real hot data that are frequently updated are not separated from the infrequently updated ones.

The double hot/cold clustering scheme [10] upgrades the hot/cold clustering scheme by regarding the first-written data as cold data because the first-written data tend to be cold. It uses two working blocks; the first-written data are moved to the cold block, together with the valid data of the victim blocks. The updated data are written to the hot block. This double hot/cold clustering scheme demonstrates a better performance than the hot/cold clustering scheme. However, the real hot data are still not separated from infrequently updated data, and the first-written hot data are mixed with cold data in the cold block, which increases the invalidation ratio of the cold blocks.

3 Proposed data-clustering scheme

In order to separate data with different update frequencies, this work proposes a new hot/cold clustering scheme that classifies data into three groups—hot, warm, and cold—according to their update frequencies. Whereas existing schemes only separate rarely updated data from the others, the proposed scheme also separates the frequently updated data, to further skew the invalidation ratio over all the blocks. The update frequency is evaluated by an update interval, which is the elapsed time since the last write.

Fig. 1 illustrates how the update frequency is evaluated and data types are dynamically transitioned. On a write request, if the update interval is shorter than a predefined threshold, the requested data are regarded as hot. If the update interval is longer than the threshold, the data are evaluated as warm because they have a long

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

update frequency. Like the existing schemes, the valid data of the victim block are regarded as cold at the garbage collection because they have not been updated since they were last written. Finally, for the first-written data that do not have an update interval, evaluating the update frequency is deferred until they are updated, or evaluated as cold at the garbage collection. Consequently, data types dynamically change according to their update interval.

In the data-type transition, the cold data cannot be changed to hot because the cold-data updates are all long updates. The time from when the cold data were last written to when they were evaluated as cold at the garbage collection is longer than the threshold. Thus, the updates of cold data are always long updates.

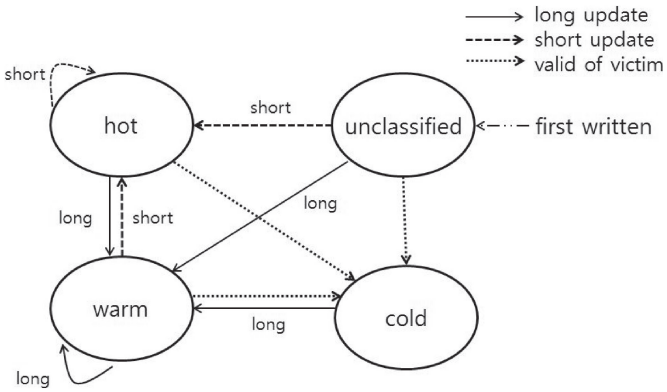


Fig. 1. Data-type transitions from when the data are first written

After evaluating the update frequency, we cluster heterogeneous data into different blocks to skew the invalidation ratio. For this purpose, the proposed scheme maintains four working blocks—hot, warm, cold, and unclassified. The updated data are written to the hot block or to the warm block according to the evaluation result, and the first-written data are clustered to the unclassified block. The valid data of the victim blocks are copied to the cold block at the garbage collection. If one of these working blocks becomes full, a clean block is allocated. If there is no clean block, garbage collection is initiated to reclaim one.

Meanwhile, in order to calculate an accurate update interval, the last update time should be recorded in the page-mapping table, which leads to an increase in the page-mapping table size and, as a result, memory consumption. To avoid this overhead, this work uses an approximated update interval: the distance between the page index where the data were most recently written, and the current clean page index where the data will be written.

For example, as seen in Fig. 2, if sector 0 (S0) was last written to page index 0 (P0) of the previous hot block and the clean page index of the current hot block is P1, then the update interval—the distance between the two indexes—is five, where a block consists of four pages. If this distance is shorter than the threshold, updated data are written to the clean page index (P1) of the current hot block; otherwise, they are written to the clean page index of the warm block. The update interval for the warm and unclassified data is similarly calculated.

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

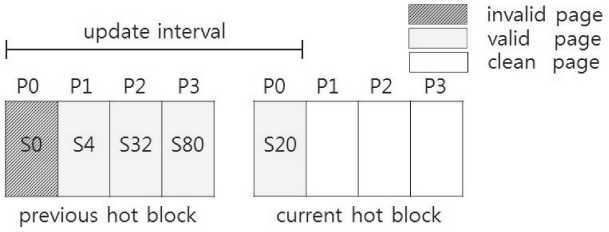


Fig. 2. Example of calculating the update interval for hot data

4 Performance evaluation

To evaluate the effect of the proposed scheme, we used an OpenSSD board called Jasmine [11]. The Jasmine board embeds the original page-mapping FTL that uses the greedy replacement scheme; implementing other FTL schemes is also possible. Thus, we implemented the proposed scheme, as well as the existing hot/cold clustering schemes in the board for a performance comparison. In the proposed scheme, the number of pages in a block is used for the threshold that distinguishes between a short update and a long update.

Table I shows the Jasmine board configuration. It consists of an Indilinx Barefoot SSD controller, including a 16/32-bit ARM7TDMI-S RISC microprocessor, 96 KB SRAM, 64 MB SDRAM, and 64 GB NAND flash memory. The NAND flash memory comprises eight NAND flash banks, each of which is 8 GB in capacity. A physical page is 4 KB in size, and eight pages compose a single clustered page, which is the mapping unit between the logical address space and the physical address space. A clustered block includes 128 pages. A SATA 2.0 host interface (3 Gbps) is supported for data transmission between the host computer and the board.

Table I. Jasmine board configuration

Configuration	Jasmine Board
SSD controller	Indilinx Barefoot
Microprocessor	ARM7TDMI-S RISC
SRAM	96 KB
SDRAM	64 MB
SATA	2.0 Host Interface (3 Gbps)
NAND capacity	64 GB
Number of banks	8
Physical page size	4 KB
Clustered page size	32 KB
Pages per block	128

Using the Diskmon tool, we collected two I/O traces from PCs that were installing programs, updating Windows, browsing the Internet, editing documents, etc. We named the trace file using the operating system name and the partition size. For example, the winXP_59GB trace was collected in the 59 GB partition where Windows XP was installed. Table II shows the detailed information of the traces.

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

Table II. Trace information

Trace	Partition Size	Total bytes read	Total bytes written
winXP_55GB	55 GB	147 GB	153 GB
winXP_59GB	59 GB	239 GB	467 GB

To evaluate the performance of each FTL scheme, we replayed the traces in the Jasmine board and measured the total elapsed time. Before performing each experiment, the board was initialized in a factory mode, which erased all the NAND blocks and cleaned up the mapping table to eliminate the influences of the previous experiments. For an explicit comparison, we normalized the elapsed time for each FTL scheme, when the result of the original page-mapping scheme was scaled to one. Fig. 3 shows the result. In the legend, hot/cold denotes the existing hot/cold clustering scheme that regards valid pages of victim blocks as cold and separates them from the others [9]. Double hot/cold denotes the scheme that upgrades the hot/cold scheme by regarding the first written data as cold [10].

As seen in the figure, the proposed scheme delivers the best performance for both traces. Compared to the original page-mapping scheme, it reduces the total elapsed time by 34.5% in winXP_55GB and by 20.0% in winXP_59GB. The improvement to the hot/cold scheme is 16.1% and 10.7%, respectively. Compared to the double hot/cold scheme, our scheme reduces the elapsed time by 7.6% and 7.0%, respectively. The result indicates that classifying data into hot, warm, and cold by the update frequency and separating them from each other is effective for improving the performance.

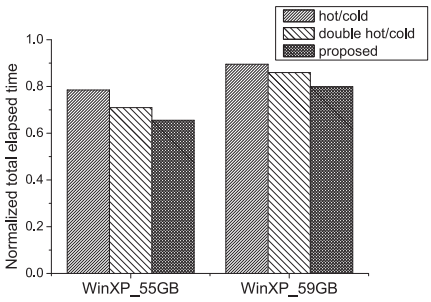
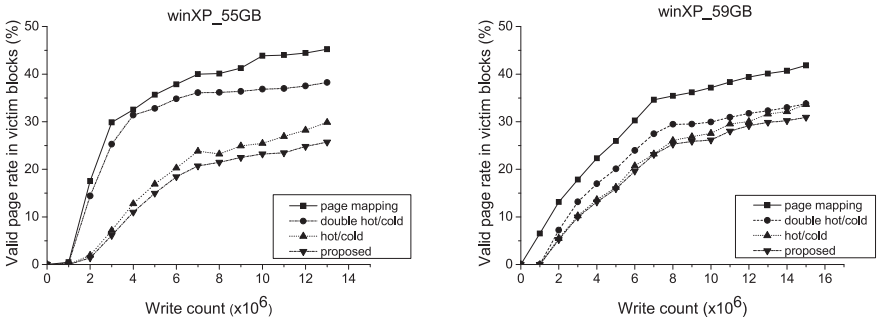


Fig. 3. Total elapsed time of each FTL scheme, when the original page-mapping result is scaled to one.

Figs. 4 and 5 explain the reasons for the performance improvements. Fig. 4 depicts the changes in the valid page rate of the victim blocks according to time in both traces. The X-axis is the sequence of write requests, namely time, and the Y-axis is the valid page rate of victim blocks at the garbage collections. Since the amount of valid data over all the blocks increases as time goes by, the number of valid pages in the victim blocks also increases. This means that the overhead of the garbage collection is increased, as the SSDs have more valid data. However, the proposed scheme slows the increase of the valid page ratio by skewing the distribution of valid data over the blocks. At the end of the trace-replaying period,

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

its valid page ratio reaches 25.7% in winXP_55GB and 30.9% in winXP_59GB, whereas that of the original page-mapping scheme is 45.2% and 41.8%, respectively.



(a) winXP_55GB (b) winXP_59GB

Fig. 4. Valid page rate of victim blocks according to time

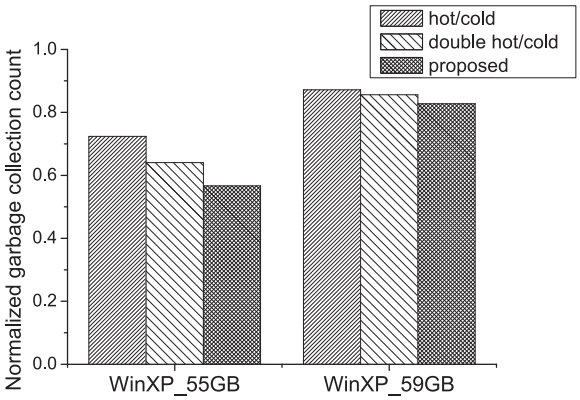


Fig. 5. Total garbage-collection count of each FTL scheme, when the result of the original page-mapping result is scaled to one.

The reduction of the valid page rate of the victim blocks results in reducing the garbage-collection frequency as well as the garbage-collection latency, because the number of reclaimed pages is inversely proportional to the valid page ratio of the victim blocks. Fig. 5 depicts the normalized total garbage-collection count of each scheme, when the result of the original page-mapping scheme is scaled to one. Thus, we observe that the proposed scheme reduces the garbage-collection count compared to the original page-mapping scheme by 43.4% in winXP_55GB and by 17.3% in winXP_59GB. The improvements to the double hot/cold scheme, which is the second best, are 5.9% and 4.1%, respectively. These reductions to the garbage-collection count and the garbage-collection latency led to the overall performance improvement.

5 Conclusion

In order to generate a skewed validation ratio over the blocks and thus reduce the garbage-collection overhead of SSDs, this work proposed classifying data into hot, warm, and cold according to their update frequencies, and clustering them into

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016

different blocks. To evaluate the update frequency without additional memory overhead, an update interval that was the distance to the last-written index was used.

The performance evaluation conducted on an OpenSSD board showed that the proposed scheme reduced the total elapsed time by 34.5%, compared to the original page-mapping scheme, and by 7.6% compared to the existing hot/cold clustering schemes. The improvement was achieved by reducing the garbage-collection overhead. The reduced validation ratio of the victim blocks led to a reduction in both the garbage-collection latency and its frequency.

Acknowledgments

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A2011586).

DOI: 10.1587/elex.12.20150984
Received November 19, 2015
Accepted November 27, 2015
Publicized December 16, 2015
Copyedited January 10, 2016