International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 15 (2017) pp. 4836-4839

Improving SSD Simulator for High Reliability of Performance Evaluation

J. Kim[1], M. Park[2] and I. Shin[3,*]

Under graduate Student [1], Under graduate student [2], Associate Professor [3]
[1,2,3] Department of Electronic Engineering, Seoul National University of Science & Technology, Nowon-gu, Seoul, South Korea.

**Corresponding author's*

Abstract

As the SSD market grows, research is underway to improve the performance and endurance of SSDs. For this research, SSD simulator is needed. SSDsim is a SSD simulator that complements the disadvantages of existing SSD simulators and complies with advanced commands and restrictions. However, there is a different part in setting up the initial environment of the simulator, compared to the actual SSD environment. In this study, the SSDsim simulator is improved by modifying it to be similar to the actual SSD environment and adding new parameters such as valid page ratio.

Keywords: Performance evaluation, simulator, SSD

INTRODUCTION

NAND flash memory based SSDs(solid state drive)s have received much attention over the past few years. As prices are decreasing, the use of SSD has greatly increased, and there are many studies that improve the performance and endurance of SSDs.

To evaluate the performance, SSD simulator were used in the previous researches. There have been several open source SSD simulators [1-2], but most of them do not support the advanced commands or they do not comply with the restrictions of using advanced commands.

Hu. et el. implemented an SSD simulator called SSDsim that complements the disadvantages of the existing open source SSD simulators and uses the advanced commands [3]. However, SSDsim has a problem in that the accuracy of the simulation is degraded by making different assumptions from the actual SSD in the process of setting the initial environment of the SSD. In this study, we proposed a method to improve the accuracy of the simulation.

BACKGROUND

NAND flash memory consists of blocks and pages. A block is the basic unit of an erase operation and consists of several pages. Pages are the basic unit of read/write operations. Because NAND flash memory does not support the overwrite operation, an out-of-place update, which writes data to a new location instead of the original location and invalidates the obsolete data, is performed on NAND flash memory. Thus, devices using NAND flash memory embeds a flash translation layer (FTL) to perform the out-of-place update. FTL maintains a mapping table to remember the current location of the valid page.

FTL schemes are classified to page mapping [4], block mapping [5], and the hybrid mapping [6] according to the mapping unit. Even though the page mapping requires large memory to maintain the mapping table, it is generally used for the SSDs because it delivers the best performance. The SSDsim also implements the page mapping FTL.

The page mapping scheme writes data in page units. When a write request is received, a logical page number (LPN) is calculated from the sector number. With the LPN, it searches for the mapping table, and if the previous entry exists, the previous physical page is invalidated. Then, it finds a clean page and writes data to the page. Then, the mapping table is updated.

As the write operation accumulates, clean pages will be exhausted and a garbage collection to reclaim clean pages is performed. First, a victim block to reclaim is selected. Generally, the mostly invalidated block is selected because it incurs the least overhead of performing the garbage collection. Then, the valid pages of the victim block are copied to an extra block that is reserved for the garbage collection. The victim block is erased and becomes a new extra block. The previous extra block becomes a new working block. The garbage collection process takes a long time because it incurs multiple read/write operations and a block erase operation. The latency of the garbage collection process is calculated as (1),

$$T_{gc} = N_{vp} * (T_{read} + T_{write}) + T_{erase} \quad (1)$$

where N_{vp}, T_{read}, T_{write}, and T_{erase} represent the number of valid pages, the time required for the page read operation, the time required for the page write operation, and the time required

for the block erase operation, respectively. Therefore, as the victim block has more valid pages, the latency of the garbage collection becomes longer [7].

Modifying SSDsim simulator

The performance of an SSD dramatically varies depending on whether the SSD is a clean SSD or an aged SSD. The aged SSD means that data are already written and thus the clean pages may not be sufficient which can trigger the garbage collection. On the contrary, in the clean SSD all the blocks are clean and thus the garbage collection is delayed to some extent. Thus, the performance of the aged SSD is much lower than that of the clean SSD.

The SSDsim, an open source SSD simulator, set the initial environment of SSD with the parameter that is called the aged ratio, which represents the percentage of pages already written. If the value of this variable is set to zero, the SSD is the clean SSD. Otherwise, the SSD is the aged SSD, and we can configure the degree of the aging.

The problem is that the aged SSD is quite different from the actual SSD environment. In the SSDsim, if the aged ratio is set, all the blocks have the same number of clean pages as shown in fig. 1. For example, if the ratio is 0.6 and the block consist of 5 pages, three pages of each block are already written and invalidated, and the two remaining pages are clean. However, in reality, the page mapping FTL does not balance all the clean pages in the blocks. Instead, it uses the working block. Once a clean block is chosen for the working block, it is used for handling the subsequent write requests until the clean pages are exhausted. If the working block is fully written, it is not used any more until the garbage collection process reclaims it, and another clean block is used for a new working block. Thus, the actual aging is shown like fig. 2. As seen in the figure, some blocks that were used for the working block are fully written, and the current working block has several clean pages, while the remaining clean blocks are all clean.

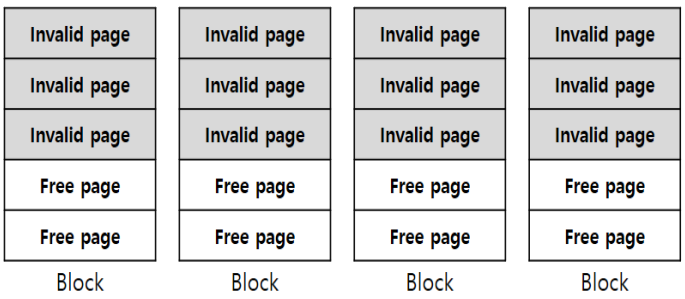

Figure 1: Aging in the original SSDsim

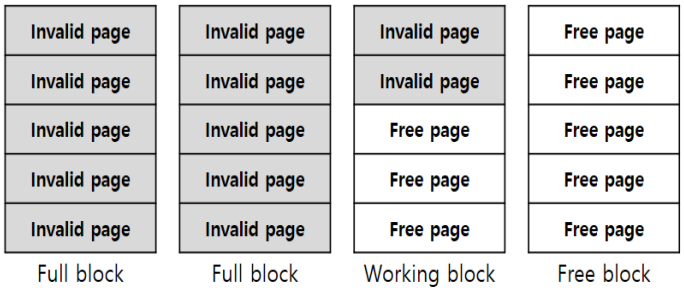

Figure 2: Aging in the actual page mapping SSD

In order to address this problem, we modified the SSDsim simulator so that the aging process is performed like the actual SSD. If the aged ratio is 0.5, roughly a half of the entire blocks are fully written, where as the other half is clean. The working log block has a half number of clean pages.

Another problem of the SSDsim aging is that all written pages are invalidated as seen in fig. 1 and fig. 2. There is no valid page. However, in reality, there are valid pages in the aged SSDs and the latency of the garbage collection is quite different according to the number of the valid pages as seen in (1). Therefore, the performance evaluation result of the original SSDsim does not assure the correctness.

In order to solve this problem, we add a new parameter, valid page ratio, to set the ratio of valid pages. If this value is set to zero, all the written pages are invalidated like the original SSDsim. Otherwise, fully written blocks have different number of valid pages, in other words, validation ratio. We randomize the validation ratio of each block. In other words, the validation ratio of each block may differ from each other, but the average of the valid page ratio is the same with the parameter. The modified aging is shown in fig. 3

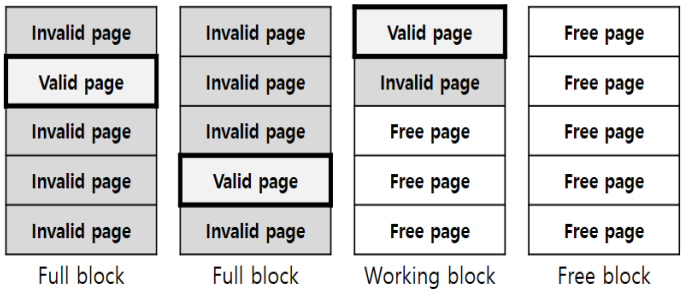

Figure 3: Aging that adds the valid page ratio

EXPERIMENT RESULTS

We analyzed the influence of the modified aging configuration on the overall SSD performance. Two groups of server traces were used: the MSRC trace files (wedv_0, src2_0, and ts_0) were collected from the servers of the MSRC [8], and the financial1 trace was downloaded from the

SPC website [9]. The detailed trace attributes are shown in table 1.

Table 1: Trace Attributes

Group	Trace	# of Request	Write Ratio (%)
MSRC	src2_0	449	89
	ts_0	388	82
	wdev_0	529	80
SPC	financial1	8	77

The page mapping scheme was used as the FTL scheme, and the dynamic allocation scheme that determines the target plane of the write requests considering the current state of channel and NAND flash chip was used. The advanced features such as the die interleaving and multiplane operations were turned off. The parameter, aged ratio is set to 0.8, and the valid page ratio was varied between 0.0 to 0.3. Other parameters of the simulator were not changed. Table 2 shows the SSD specification of the SSDsim simulator including the NAND flash memory performance features.

Table 2: Trace Attributes

Parameters	Values
Number of channels	2
Number of chips per channel	4
Number of dies per chip	2
Number of lanes per die	2
Number of block per plane	2048
Number of page per block	64
Page size	2KB
Page read to register	20µs
Page write from register	200µs
Block erase	1.5ms

Fig. 4 shows the average response time of write requests. The x-axis represents each trace, and the y-axis represents the average response time. For clean comparison, we calculated the relative value assuming that the performance of the original SSDsim is one. The basic denotes the original SSDsim, and approach1 denotes the modification so that the number of clean pages are different over blocks, in other words, some blocks are fully written while the others are clean and the working block has several clean pages. Approach2 denotes the modification that adds the valid page ratio. The number is the valid page ratio, which is varied from 0.0 to 0.3. Because approach1 + approach2 (0.0) is the same with the approach1, their evaluation results are the same.

The results show that applying the approach1 slightly reduces the average response time on all the traces. The reduction is conspicuous in ts_0. This is because the number of clean pages are different over blocks and there are clean blocks which delays the initiation of the first garbage collection process. Once the garbage collection is first triggered, its frequency is larger than the original SSDsim simulator.

On the contrary, when we apply the valid page ratio (approach2), the overall performance is significantly degraded. The degrade is more severe when the valid page ratio increases. When the valid page ratio increases from 0.1 to 0.3, the response time increases approximately two times on the most traces. This shows that the valid page ratio is crucial when configuring the aging process, and the sustained performance of SSDs may be quite low compared to that of the clean SSDs.

Fig. 5 shows the total block erase count. For clean comparison, we calculated the relative value assuming that the performance of the original SSDsim is one. The results are very similar with fig. 4. When applying the approach1, the erase count slightly reduces on all the traces. The reduction is conspicuous in ts_0. This is because the initiation of the first garbage collection process was delayed. When applying the valid page ratio (approach2), the erase count is dramatically increased. When the valid page ratio increases from 0.1 to 0.3, the erase count increases more than 30% on all the traces.

Conclusively, the original SSDsim simulator did not reflect the characteristics of the real SSDs. Especially, it did not consider the valid page ratio in the aging process, which significantly influences on the overall performance. The influence of unbalancing the distribution of clean pages over blocks was less crucial.

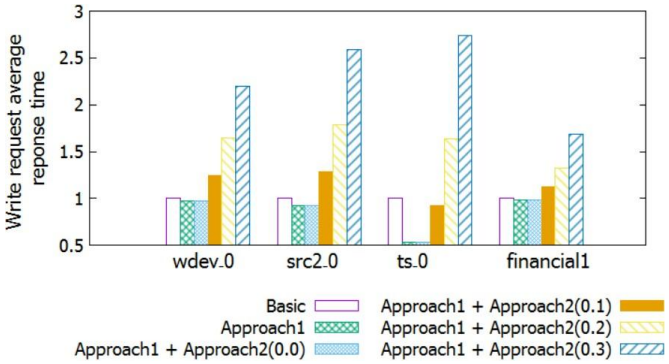

Figure 4: Average response time of write requests

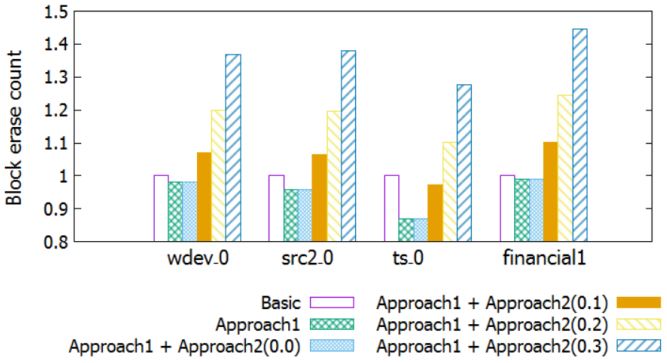

Figure 5: Total block erase count

CONCLUSION

In this paper, we proposed to modify the aging process of the SSDsim simulator to reflect the aging process of the real SSDs. The original simulator had the two problems. First, it levels the number of clean pages over the blocks. However, it is different from the real SSDs. In the real SSDs, some blocks are clean while the other blocks are fully written. The working block has several clean pages. Second, all the written pages are invalidated in the original SSDsim simulator. However, in reality, several written pages are valid, which influences the performance of the garbage collection. Furthermore, the valid page ratio differs over the blocks. In order to address these problems, we modified the aging process of the simulator so that it resembles the real SSDs. The performance evaluation results showed that the modifications influenced on the overall performance significantly. Especially, the valid page ratio severely degraded the overall performance. This implies that the performance evaluation results with the original SSDsim simulator may not be realistic and practical. We expect that the modified simulator will be used to produce the trustful performance evaluation results of the various schemes including FTL, buffer management, and page allocation to enhance the SSD performance.

ACKNOWLEDGEMENTS

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-2016R1D1A1A09918559).

REFERENCES

[1] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, "FlashSim: A Simulator for NAND Flashbased Solid-State Drives," Technical Report CSE-09-008, 2009.

[2] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," USENIX Annual Technical Conference, pp. 57-70, June 2008.

[3] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity," International conference on Supercomputing, pp. 96–107, 2011.

[4] A. Ban, "Flash file system," United States Patent. No. 5,404,485, April 1995.

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

[6] 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.

[7] I. Shin, "Hot/Cold Clustering for Page Mapping in NAND flash memory", IEEE Trans. Consumer Electron., vol. 57, no. 4, 2011.

[8] Microsoft Research Center, "MSRC I/O traces," ftp://ftp.research.microsoft.com/pub/austind/MSRC-io-traces

[9] Storage Performance, "SPC I/O traces," http://skuld.cs.umass.edu/traces/storage/Financial1.spc.bz2