International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 1 (2019), pp. 119-123
© International Research Publication House. http://www.irphouse.com

Evaluating Performance Degradation due to Block Size Increase in 3D NAND-based SSDs

Ilhoon Shin

*Associate Professor, Department of Electronic Engineering,
Seoul National University of Science and Technology, South Korea.*

(Orcid: 0000-0003-2819-0398)

Abstract

Recently, 3D NAND has been proposed that dramatically improves the integration density of NAND flash memory. 3D NAND has advantages over conventional 2D NAND in that it reliably stores data, improves the integration density, and speeds up data writing, but there is a potential risk that the NAND block size increases significantly. When the block size increases, the number of valid pages of the victim block increases during garbage collection, which increases the latency of the garbage collection. As a result, the SSD performance may deteriorate. Therefore, in this study, the performance degradation due to the increase of the block size in the 3D NAND-based SSD is analysed and the following conclusions are drawn. First, the average performance is not degraded much. The average response time is maximally increased up to 7.1% in fin1 trace. This is because the latency of individual garbage collection is longer, but the total number of garbage collections is reduced. Second, the decrease in the tail performance is relatively large. fin1 trace have an increase of about 4.8x tail latency, and other traces have a longer latency of 5.6% - 20.2%. Therefore, when using 3D NAND, we should focus on improving the tail performance rather than the average performance, and it is expected that the performance enhancement through clustering hot data and cold data separately is relatively large in 3D NAND. In other words, the block size increase of 3D NAND can be an opportunity for improving SSD performance.

Keywords: 3D NAND, SSD, Garbage collection, Block size

I. INTRODUCTION

NAND flash memory has been used as a storage medium for USB memory, SD card, eMMC, and solid state drives due to its lightweight, low energy consumption, and high read speed. In recent years, the integration of NAND flash has dramatically improved due to the continuous development of semiconductor technology and the appearance of MLC (multi-level cell), TLC (triple-level cell), QLC (quadruple-level cell), where one NAND cell can store multiple bits . As a result, SSDs that use NAND flash memory as storage media replace hard disks in the server and enterprise storage markets as well as laptops. While the storage capacity of NAND flash have been greatly improved, its performance and stability have deteriorated. As the size of the cells becomes smaller and the distance between the adjacent cells becomes closer to each other, it is more likely that the written values are unintentionally changed due to the natural loss of electron and due to the interference between the adjacent cells. Also, as one cell expresses several bits, the probability that the value is erroneously written increases. As a result, the write latency of NAND flash memory has been slower for the precise control of electron injection into cells [1, 2].

Recently, 3D NAND for stacking NAND flash cells vertically has been proposed as a way to increase the storage capacity while mitigating the side effects due to the small cells and the close distance between cells. Conventionally, cells are constructed only in a plane, whereas in 3D NAND, cells can be stacked vertically, resulting in a dramatic improvement in storage capacity. Since the distance between adjacent cells in the same plane and the cell size can be increased, it is more stable than 2D NAND and provides a faster write operation [3, 4]. However, 3D NAND has the disadvantage that the block size increases compared to 2D NAND. A block is the unit of an erase operation. If the block is larger, the latency of the individual garbage collection is increased, and thereby the SSD performance may be degraded [3, 4]. Previous researches evaluated the performance degradation by using 3D NAND with SSD simulators and based on the result proposed a sub block erase that deletes only a part of the block instead of the whole block when performing garbage collection [3, 4]. However, these studies have limitations that they employed hybrid mapping FTL (flash translation layer) that delivers a low performance instead of page mapping [3], that a static die binding that restricts the internal parallelism of SSDs is used instead of dynamic die binding [3], and that the block size of the 3D NAND is set to be similar to 2D NAND [4].

Therefore, this paper aims to evaluate the performance degradation of SSDs due to block size increase when 3D NAND is used. We use the SSDSim simulator [5] that employs the dynamic die binding and the page mapping FTL. Other SSD and NAND parameters are set referring to the previous researches.

The remainder of the paper is organized as follows. Section 2 discusses the background and the related works. Section 3 presents the simulation setup and evaluation results. Finally, section 4 concludes the paper and discusses future work.

II. BACKGROUND AND RELATED WORK

NAND flash memory consists of blocks and pages. A block is a basic unit of an erase operation, and a page is a basic unit of a read/write operation. In 2D NAND, a page is usually 4-8 KB in size, and a block consists of 128 or 256 pages. Since NAND flash memory does not support an overwrite operation, the block devices that embed NAND flash memory emulate the overwrite with an out-of-place update through FTL. In other words, when a write request arrives, FTL finds a clean page where no data has yet been written, writes the data to the clean page, and invalidates the obsolete page where old data is written. Therefore, the location of data changes on every update and FTL needs to manage the mapping information between the logical address and the real address. According to the mapping unit of the address, FTL is classified into page mapping, block mapping, and hybrid mapping. Page mapping FTL using pages as an address mapping unit is commonly used in SSDs because it delivers a good performance [5-7]. The block mapping and hybrid mapping methods have the advantage of reducing the mapping table, but because of their low performance, they are mainly used in USB memory or memory cards with the small internal memory.

Meanwhile, clean pages will eventually be short by the continuous out-of-place updates, which initiates the garbage collection to reclaim clean pages. The garbage collection selects a victim block, copies all the valid pages of the victim block to another block, and erases the victim block to reclaim clean pages. Therefore, as the number of valid pages is less in the victim block, more clean pages are reclaimed, and the garbage collection latency is also shorter [8]. Therefore, when selecting a victim block, it is common to select a block with a small number of valid pages [8]. In order to reduce the number of valid pages of the victim block, some researches proposed to separate hot data, which are frequently modified, and cold data, which are rarely modified, and to store them in different blocks [9, 10].

Meanwhile, 3D NAND is known to increase the block size compared to 2D NAND. As the block size increases, the speed of erase operation is somewhat slower than that of 2D NAND [3, 4]. In addition, when garbage collection is performed, the number of valid pages remaining in the victim block is also likely to increase. Therefore, it takes a long time to perform garbage collection, which may degrade the SSD performance. However, since the number of clean pages to be reclaimed through an individual garbage collection is expected to increase, the number of garbage collection execution is expected to decrease, which contributes to improving the average performance. As a result, when 3D NAND is used, performance degradation factor and performance enhancement factor coexist. Therefore, it is necessary to analyse the performance of the 3D NAND-based SSD in the environment close to the real SSDs.

Meanwhile, SSDs connect several NAND flash chips in parallel to achieve high storage capacity and high I/O performance. A chip is composed of multiple dies, each of which consists of multiple planes. A die is a basic unit that can perform NAND operation independently, and the planes belonging to the same die can perform the same kind of NAND operation at the same time. Therefore, in order to utilize the internal parallelism, SSDs split the I/O requests into sub-requests in a page unit and distributes them in parallel to multiple chips, dies, and planes. In case of read requests, the target chip, die, and plane are fixed, and therefore the read request can be concentrated on a specific chip, die, and plane. However, since the write request is processed in an out-of-place manner, the target chip, die, and plane can be dynamically determined. Therefore, various binding policies have been proposed to maximize the internal parallelism of SSDs. [7] holds the request queue per die to determine the target die with the shortest queue length, and [5] defines the idle chip and die as the target die considering the state of each chip and die. Also, [6, 7] proposed a method to maximize internal parallelism through multi-plane operation. All of these are dynamic binding policies and perform better than static binding policy that determines the target chip, die, and plane using sector numbers [5, 7]. Therefore, performance evaluation of 3D NAND should be performed in an environment using dynamic binding policy.

III. PERFORMANCE EVALUATION

III.I Configuration

We use the SSDSim simulator [5] which implements the internal parallelism of the SSD. The simulator uses a page mapping FTL and uses state-based die binding policy [5] and multiplane operation [6]. In other words, a write request is sent to an idle chip, a die, and a plane. If there is no idle chip, a write request waits in the queue. The model SSD is structured to have four chips that are connected through two parallel channels. Each chip is composed of two dies, and each die is composed of two planes. The garbage collection is performed in the background when the number of clean pages in the plane falls below 20%.

The physical properties of 3D NAND are assumed as shown in Table 1 referring to the values used in previous studies [3, 4]. The page size is fixed at 8KB, and the block size is varied from 1MB to 6MB. As the block size increases, the block erase latency also increases. Since there is no accurate model for the relation between the block size and the erase latency, the erase latency is increased by 0.1 ms for each 1 MB increase in the block size, referring to the values used in the previous studies [3, 4].

We used four server traces (hm0, prn0, proj0, and mds0) downloaded from Microsoft Research Centre (hm0, proj0, and prn0) and two server traces (fin1 and fin2) downloaded from Storage Program Council as input traces for performance evaluation. Because the address space size of each trace is very different, the total capacity of the model SSD is set to match the address space size of the trace with the overprovision ratio of 30%. Exceptionally, the overprovision ratio for the fin2 trace is set to 80% because its address space is very small, less than 1GB of trace space.

Meanwhile, SSDs show very high performance initially because the garbage collection is not performed when all pages are clean. However, as the number of clean pages decreases and the garbage collection is performed, the performance is degraded. Therefore, the trace is repeated until the garbage collection occurs, and the performance is evaluated in the next iteration.

Table 1. 3D NAND properties

Page size	8KB
Block size	1MB, 2MB, 3MB, 4MB, 5MB, 6MB
Page read latency	49us
Page write latency	600us
Block erase latency	4ms, 4.1ms, 4.2ms, 4.3ms, 4.4ms, 4.5ms, 4.6ms

III.II Evaluation results

Fig. 1 shows the average number of valid pages of a victim block when performing garbage collection. The more valid pages, the longer it takes to copy them to other blocks, thus slowing down the garbage collection. The x-axis represents the type of 3d NAND. 3d1 means 1MB block size and 3d2 means 2MB block size. The y-axis is the average number of valid pages in the victim block. The results show that as the block size increases, the average number of valid pages increases in the most traces. Copies of valid pages are prominent in fin1 and fin2 traces, and no page copy occurs at mds0 and prn0 traces at all. In those trace, even if the block size increases to 6MB, the victim blocks are always fully invalidated.

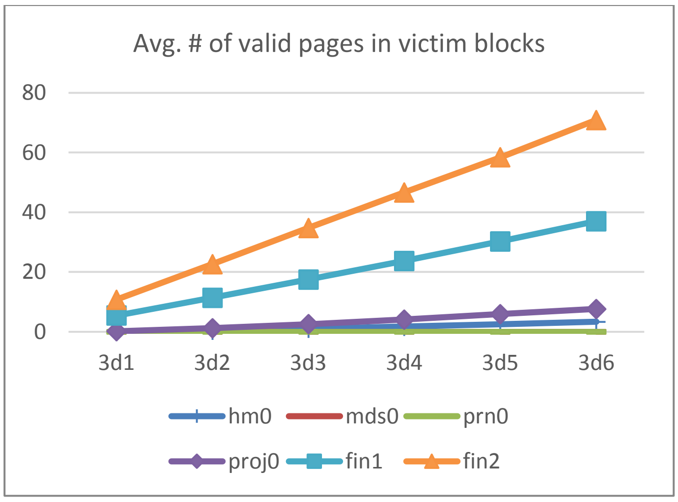

Fig. 1. Average number of valid pages in victim blocks

Fig. 2 shows the maximum number of valid pages of a victim block. The results are not significantly different from those in Fig. 1. As the block size increases, the maximum number of valid pages increases in most traces. Copying of valid pages is noticeable in fin1 and fin2 traces.

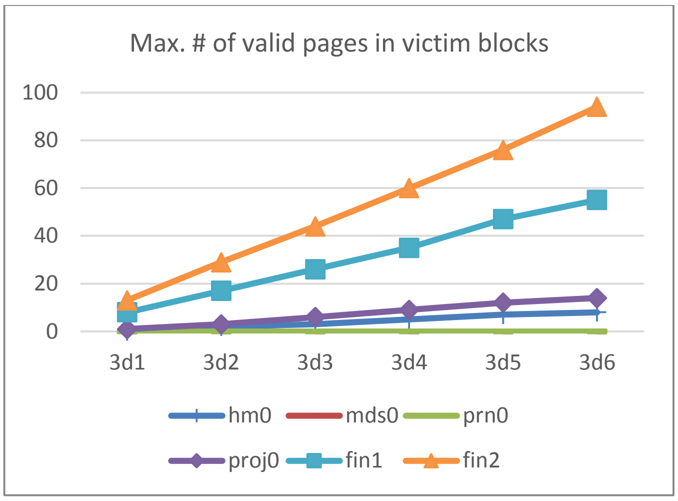

Fig. 2. Maximum number of valid pages in victim blocks

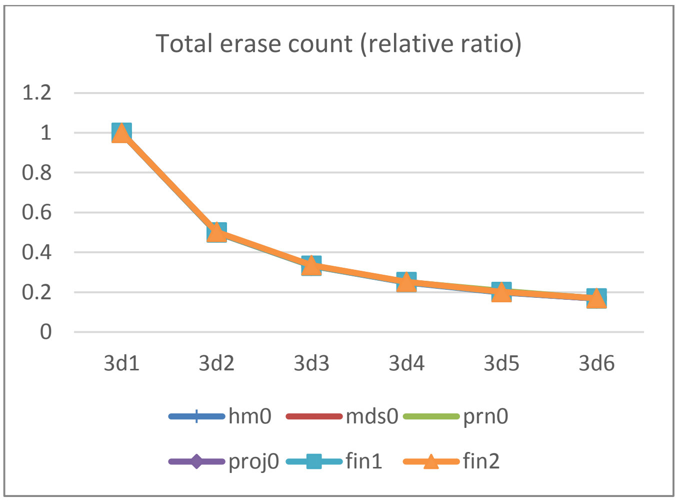

Fig. 3. Total erase count

Fig. 1 and Fig. 2 show that as the block size increases, the average and the worst-case latency of the garbage collection increase. However, as the block size increases, the number of clean pages to be reclaimed through a single garbage collection also increases, so that the number of the garbage collection execution may be reduced. Thus, fig. 3 shows the number of block erasures due to the garbage collection. Since the number of block erase is very different according to the traces, the relative value is calculated when the result of 3d1 is regarded to 1 in each trace. The figure shows that the number of block erase decreases greatly as the block size increases. The decreasing ratio is similar in all the traces. In 3d6, the number of block erase decreases to less than 20%. That is, although the average latency of the garbage collection is prolonged, the number of garbage collection executions is reduced, and the performance degradation is somewhat mitigated.

Fig. 4 shows the average response time of I/O requests. Since the response time is very different according to the traces, the relative value is calculated when the result of 3d1 is regarded to 1 in each trace. The figure shows that even if the block size increases, the degradation of the average performance is not severe. When the block size increased from 1 MB to 6 MB, the performance degradation is the largest at the fin2 trace and is about 7.1%. Proj0, fin1 and hm0 have a decrease of about 6.3%, 4.9% and 4.4%, respectively. Mds0 shows little performance change, and prn0 improves the average performance by about 2.9% when the block size increases to 6 MB. In conclusion, even if the block size increases 6 times, the average performance degradation is not serious. That is, the latency of the individual garbage collection is lengthened, but the garbage collection frequency is also reduced, so the average performance deterioration is not large.

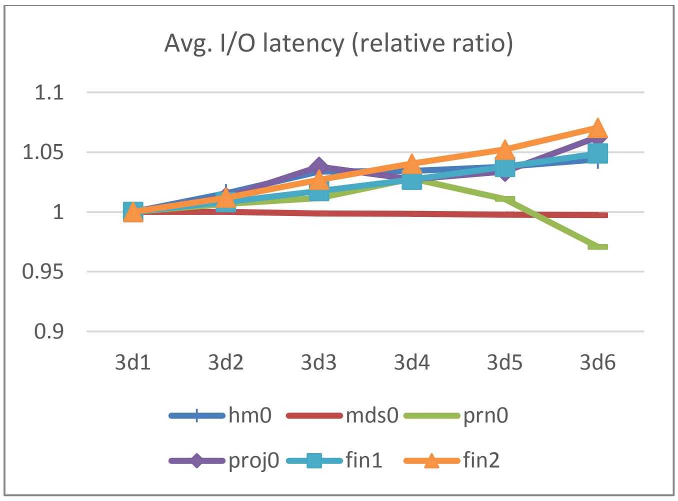

Fig. 4. Average I/O latency

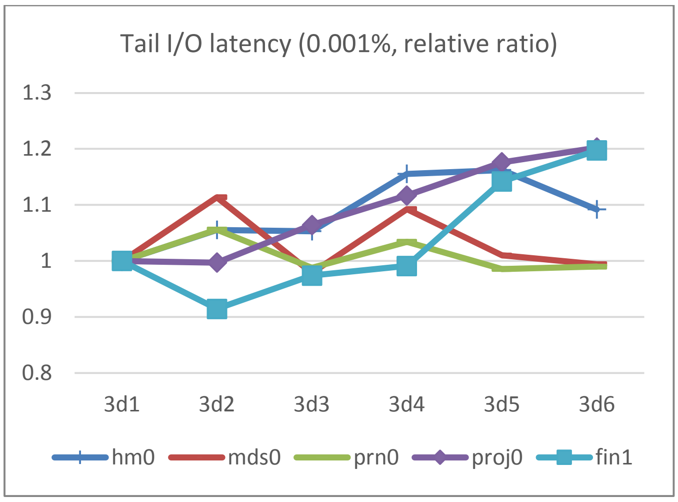

Fig. 5. Tail I/O latency (0.0001%)

Lengthening the latency of the individual garbage collection can have a significant impact on the tail performance. Fig. 5 shows the average response time of the lower 0.001% I/O requests when the total I/O requests are sorted in ascending order by the response time. Since the response time is very different according to the traces, the relative value is calculated when the result of 3d1 is regarded to 1 in each trace. The results show that as the block size increases, the tail latency tend to increase in overall and the increase is larger than that of the average performance. Fin1 trace, which is excluded from the graph because the increase is too large, has the increase of 4.8x. Proj0 shows a maximum degradation of 20.2% and fin1 has a 19.7% degradation when the block size is increased to 6 MB. Hm0 trace has the largest performance degradation (16.2%) when the block size is 6 MB. The degradations of mds0 and prn0 are relatively small. In the both traces, the maximal degradations occur when the block size is 2MB, which are 11.4% and 5.6%, respectively. Conclusively, the degradation of the average performance is not large. However, the tail performance deteriorates remarkably especially in fin1 trace as the block size increases.

IV. CONCLUSION

This work evaluated the performance degradation due to block size increase of 3D NAND in high-end SSDs that employ the page mapping FTL and maximize the internal parallelization. As a result, unlike the conclusions of previous studies, the average response time of I/O requests did not increase significantly. The maximum degradation was 7.1% in fin1 trace. This is because the latency of the individual garbage collection is longer, but the number of garbage collection executions is reduced also. However, the tail performance decreased significantly. In fin1 trace, the tail latency increased by about 4.8x, while proj0 and fin1 traces had the increase of 20.0% and 19.7%, respectively.

In conclusion, when using 3D NAND, we need to focus on improving the tail performance rather than the average performance. Also, the increase in the block size seems to be an opportunity for the performance improvement because the garbage collection latency reduction will be greater by hot and cold data separation. We plan to apply the separation policy of hot data and cold data to 3D NAND based SSDs considering the internal parallelization.

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] R. Liu and C. Yang, “Optimizing NAND flash-based SSDs via retention relaxation", USENIX FAST Conference, Article. 14, Feb. 2012.

[2] I. Shin, "Applying fast shallow write to short-lived data in solid-state drives", IEICE Electronics Express, 2018:15(13):1–9.

[3] C. Liu, J. Kotra, M. Jung, and M. Kandemir, “PEN: Design and evaluation of partial-erase for 3D NAND-based high density SSDs ", USENIX FAST Conference, Feb. 2018.

[4] T. Chen, Y. Chang, C. Ho, and S. Chen, “Enabling sub-blocks erase management to boost the performance of 3d NAND flash memory”, ACM DAC Conference, June 2016.

[5] 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", ACM International Conf. on Supercomputing, Tucson, Arizona, pp. 96–107, May 2011.

[6] I. Shin, "Improving internal parallelism of solid state drives with selective multi-plane operation", Electronics Letters, 2018:54(2):64–66.

[7] C. Park, E. Seo, J. Shin, S. Maeng, and J. Lee “Exploiting internal parallelism of flash-based SSDs,” IEEE Computer Architecture Letters, 2010:9(1):9–12.

[8] A. Kawaguchi, S. Nishioka, and H. Motoda, “A Flash-Memory based File System”, USENIX Annual Technical Conference, pp. 155-164, 1995.

[9] I. Shin, “Hot/Cold Clustering for Page Mapping in NAND flash memory”. IEEE Transactions on Consumer Electronics, 2011:57(2):1728–1731.

[10] J. Kim and I. Shin, “Clustering Data According to Update Frequency to Reduce Garbage-Collection Overhead in Solid-State Drives”, IEICE Electronics Express, 2018:13(1):1-8.