Applying fast shallow write to short-lived data in solid state drives

Ilhoon Shin

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

a) ilhoon.shin@snut.ac.kr

Abstract: In NAND flash memory, a high threshold voltage during incremental step pulse programming speeds up the write operation but reduces the data retention time, and vice versa. Current NAND flash memory uses a low threshold voltage to satisfy the industry standard, which requires data retention of more than a year, and as a result its write operation latency tends to increase as semiconductor process technology progresses. However, actual server workload analysis indicates that much of the data are short-lived and do not require a long retention time. Writing those short-lived data slowly with a low threshold voltage is inefficient. Fast write with a high threshold voltage should be employed for short-lived data and slow write should be employed only for long-lived data. Therefore, this work proposes a method that predicts the lifetime of data based on write request size and selectively applies fast write to short-lived data. The results of evaluations using representative server workloads on an SSD simulator indicate that the proposed method improves an average performance by up to 41.14% compared with the existing method. Further, the increase in total block erasures due to the wrong prediction is limited to 5.60%.

Keywords: NAND flash memory, solid state drives, data lifetime, shallow write

Classification: Circuits and modules for storage

References

[1] D. Narayanan, et al.: “Migrating server storage to SSDs: Analysis of tradeoffs,” EuroSys (2009) 145 (DOI: 10.1145/1519065.1519081).
[2] G. Soundararajan, et al.: “Extending SSD lifetimes with disk-based write caches,” USENIX FAST (2010) 8.
[3] R. Liu, et al.: “Optimizing NAND flash-based SSDs via retention relaxation,” USENIX FAST (2012).
[4] Y. Pan, et al.: “Quasi-nonvolatile SSD: Trading flash memory nonvolatility to improve storage system performance for enterprise applications,” IEEE HPCA (2012) 1 (DOI: 10.1109/HPCA.2012.6168954).
[5] K.-D. Suh, et al.: “A 3.3 v 32 mb NAND flash memory with incremental step pulse programming scheme,” IEEE J. Solid-State Circuits 30 (1995) 1149 (DOI: 10.1109/4.475701).
[6] L. M. Grupp, et al.: “Characterizing flash memory: Anomalies, observations, and applications,” IEEE/ACM MICRO (2009) 24 (DOI: 10.1145/1669112.1669118).

[7] K. Prall: “Scaling non-volatile memory below 30 nm,” IEEE Non-Volatile Semiconductor Memory Workshop (2007) 5 (DOI: 10.1109/NVSMW.2007.4290561).

[8] Y. Hu, et al.: “Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity,” ACM ICS (2011) 96 (DOI: 10.1145/1995896.1995912).

[9] I. Shin: “Improving internal parallelism of solid state drives with selective multi-plane operation,” Electron. Lett. 54 (2018) 64 (DOI: 10.1049/el.2017.3912).

[10] F. Chen, et al.: “Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing,” IEEE HPCA (2011) 266 (DOI: 10.1109/HPCA.2011.5749735).

[11] S. Park, et al.: “Exploiting internal parallelism of flash-based SSDs,” IEEE Comput. Arch. Lett. 9 (2010) 9 (DOI: 10.1109/L-CA.2010.3).

[12] A. Ban: Unites States Patent 5,404,485 (1995).

[13] Microsoft Research Center: ftp://ftp.research.microsoft.com/pub/austind/MSRC-io-traces/.

1 Introduction

Solid state drives (SSDs) that connect multiple NAND flash memory chips in parallel provide much higher I/O operations per second and consume less energy than hard disk drives. In addition, concomitant with advances in semiconductor technology, the density of NAND flash memory has rapidly improved, and the price per bit of SSDs has decreased significantly. As a result, SSDs are replacing hard disks in mobiles, personal computers, servers, and datacentres [1].

According to the analysis of server workloads, a considerable amount of data have a very short update period [2, 3]. In other words, short-lived data and long-lived data are mixed. Because short-lived data do not require a long retention time, the write operation for short-lived data can be accelerated by using a shallow write, which is fast but does not guarantee a long retention time [3, 4].

However, existing NAND flash memory only supports deep write mode, which is slow but guarantees a long retention time, to meet the Joint Electron Device Engineering Council (JEDEC) standard—which requires that data be maintained for more than a year. Thus, even short-lived data are written using slow deep write, which is inefficient. To overcome this problem, Liu et al. proposed a method that regards all host writes as short-lived data and writes them using shallow write, assuming that NAND flash memory supports both shallow write and deep write modes [3]. However, as their method assumes that all host writes are short-lived data, it is likely to incorrectly predict the lifetime of the data. If the data written in shallow write are not changed for a certain period, they should be rewritten in deep write for retention time guarantee. This incorrect prediction overhead can degrade the overall performance and increase the total block erasure count, resulting in reduced SSD lifetime.
This paper proposes a method that predicts the lifetime of data, and consequently writes short-lived data with shallow write and long-lived data with deep write. The proposed method predicts the lifetime of the data based on the write request size. It evaluates data with a small write request size to be short-lived data because they are likely to be updated frequently, and data with a large write request size to be long-lived data. The lifetime prediction method is simple but produces good results with no additional memory overhead. The results of evaluations conducted using representative server traces on an SSD simulator indicate that, on average, the proposed method improves performance by up to 41.14% compared to the existing method that uses only deep write. Further, the increase in the total block erasure count, which is caused by rewriting overhead due to incorrect lifetime prediction is limited to a maximum of 5.60%.

The remainder of the paper proceeds as follows: Section 2 explains an overview of NAND flash memory and SSDs. Section 3 proposes a selective shallow write policy. Section 4 presents the performance evaluation results, and Section 5 draws the conclusions.

2 NAND flash memory and SSDs overview

NAND flash memory is a kind of EEPROM that expresses data through the amount of electrons charged to a floating gate of each cell [3, 4, 5, 6, 7]. For example, in a single level cell (SLC) NAND, in which one cell represents one bit, a state in which the floating gate is empty represents one, and a state in which electrons are fully charged in the floating gate represents zero. The initial state of the floating gate is empty, which is one. Since the floating gate is wrapped up with an oxide, which is an insulator, the value of one is maintained if no voltage is applied. In order to modify the value to zero, that is, to charge the floating gate, a sufficiently high voltage must be applied to the floating gate. The high voltage let electrons pass through the oxide and charged into the floating gate. This is a write operation of NAND flash memory, and the write operation is performed in a NAND page unit. Conversely, to change the value of zero back to one, the voltage is applied in the opposite direction of the write operation. The electrons in the floating gate break through the oxide film by the voltage and discharge to express the value of one. This is an erase operation, and the erase operation is performed on a NAND block unit.

Meanwhile, during the process of charging and discharging electrons, the oxide is gradually damaged, and eventually, the possibility of electrons to be naturally discharged even when the voltage is not applied increases. This error that changes its value due to natural discharge is called a retention error. As the block erase and page write operations accumulate, the endurance of the floating gate becomes worse and the bit error rate such as the retention error increases.

Meanwhile, the endurance and raw bit error rate of NAND flash memory have been continuously deteriorating as NAND integration increased [3, 4, 5, 6, 7]. As semiconductor process technology progresses, the physical size of each cell is gradually decreasing and the maximum number of electrons that can be charged in the floating gate also decreases. This means that the margin for distinguishing
between one and zero has gradually decreased. In addition, in the case of multiple level cell (MLC) NAND—in which two bits are represented with one cell—or triple level cell NAND—in which three bits are represented with one cell—it is necessary to further subdivide the state in which electrons are charged. For example, assuming that in the MLC NAND ‘11’ represents the empty state and ‘00’ is the fully charged state, the degree of charge of the electrons must be subdivided to express ‘01’ and ‘10’. Therefore, the margin for distinguishing each value is reduced, and the value is likely to change even if only a few electrons are lost owing to cell damage.

Therefore, in order to precisely control the amount of charge, the modern NAND performs write operations via the incremental step pulse programming (ISPP) method [3, 4, 5]. The ISPP method charges the cell by stepping up the voltage by the threshold voltage until the target electrons are charged. The smaller the threshold voltage is, the more accurately the charge amount can be controlled and the guaranteed retention time is increased. However, the write operation is retarded because of increased repetition. On the other hand, using a high threshold voltage speeds up the write operation but has a drawback in that the guaranteed retention time is short because the value can be changed by the natural discharge of small number of electrons. Therefore, the current NAND flash memory uses a low threshold voltage to meet the JEDEC specification which requires the retention time more than a year. As a result, the speed of the write operation has been decreasing with increasing NAND density.

Meanwhile, although the operation speed of NAND flash memory is gradually lowered, the overall performance of SSDs has been continuously improved. Performance enhancements of SSDs have been achieved primarily through their parallel processing structure [8, 9, 10, 11]. A SSD connects multiple NAND flash chips with parallel channels that can transfer data simultaneously. A NAND chip is composed of multiple dies, and each die has multiple planes. The die is the minimum unit that can perform NAND operations independently, and the planes belong to the same die can process the same type of operation simultaneously. Therefore, the response time of each I/O request can be greatly reduced by dividing it into multiple sub-requests of page size, striping sub-requests across multiple chips, dies, and planes, and processing them concurrently. In order to maximize the parallel processing capability of SSDs, intensive researches [8, 9, 10, 11] have been performed. They focus on determining the target chip and die of sub-write requests to evenly distribute them and fully utilize the processing capability. These policies that improve the parallel processing capability of SSDs can be used together with the selective shallow write policy proposed in this study to contribute to the overall performance improvement of SSDs.

### 3 Selective shallow write policy

In this study, to utilise the server trace characteristic of high short-lived data ratio, SSD firmware that operates as follows is designed. NAND flash memory is assumed to support two write modes: a shallow write with a short retention time and a deep write with a long retention time [3]. Further, the target SSDs employ a
page mapping flash translation layer (FTL) to find the physical location of sectors [8, 9, 10, 11, 12]. Read requests are processed as in the existing page mapping FTL. Upon arrival of a write request, the lifetime of the data is predicted using the request size. If the request size is below a threshold, it is predicted to be short-lived data; otherwise, it is predicted to be long-lived data. Short-lived data are written in shallow write mode, whereas long-lived data are processed in deep write mode.

To prevent shallowly written data and deeply written data from being mixed in a NAND block, FTL uses two working blocks, a shallow working block and a deep working block. Short-lived data are written to the shallow working block and long-lived data to the deep working block. If there is no space in the working block, a new clean block is allocated and used as a working block. The garbage collection for reproduction of a clean block is performed in the background when the number of clean blocks falls below the predetermined threshold, such as 10% of all the blocks.

Meanwhile, if the shallowly written data remain unmodified for a long time, it must be rewritten in deep write mode to guarantee errorless retention. For example, if a shallow write guarantees a retention time of one week, the shallowly written data should be rewritten before one week has passed after being written. For this rewriting operation, the firmware uses an additional mode bit that indicates the write mode of a block [3]. That is, if a clean block is used as the shallow working block, the mode bit is set to one. In addition, a reference bit is added to each block to indicate whether the retention time of the shallow data is about to expire [3]. When data are written in the shallow write, the reference bit is set to one, and it is periodically reset to zero every half-life of the retention time guaranteed by the shallow write mode. Therefore, all valid pages of the shallow block with reference count zero are regarded as having their retention time about to expire, and are thus rewritten in the deep write.

Fig. 1 illustrates this rewriting process in more detail. First, time is divided into multiple periods (Fig. 1(a)). Period length is the half-life of the retention time guaranteed by the shallow write. If the blocks A, B, and C are shallowly written during period $i$, their write mode and reference bits are set to one (Fig. 1(b)). A write mode bit value of one means that the block is written shallowly, and a reference bit value of one means that the retention time of the shallowly written block does not expire in the next period. At the end of each period, the firmware finds shallow blocks with a reference bit equal to one and resets their reference bits to zero (Fig. 1(c)). A reference bit value of zero means that the retention time of the block will expire in the next period. During period $i + 1$, the firmware finds shallow blocks with a reference bit equal to zero and rewrites their valid pages to other deep working blocks in the deep write mode (Fig. 1(d)). Thus, the valid pages of the shallowly written blocks in period $i$ are all rewritten in the deep write mode during period $i + 1$ before the retention time expires. To minimize performance degradation, rewriting is performed in the background whenever there are no foreground I/O requests. However, in the end of each period, if the expiration of the shallow block with reference count zero is imminent and it is not possible to continue postponing rewriting, the rewrite is processed even if there are foreground I/O requests to process.
Meanwhile, the memory overhead of using the write mode and the reference bits is not significant because two bits are required for each NAND block [3]. For an SSD having 1 TB capacity with 1 MB block size, the memory overhead is 256 KB.

4 Performance evaluation

To evaluate the performance of the proposed method, a trace-driven simulation was performed on the SSDSim simulator [8]. Four server traces were downloaded from the Microsoft research centre (MSRC) website [13] and used as input traces. Table I lists the main features of each trace.

![Fig. 1. Rewriting process of shallowly written blocks.](image)

| Trace | Read request ratio (%) | Avg. read req. size (sectors) | Avg. write req. size (sectors) | Avg. inter-arrival time (ms) |
|-------|------------------------|-------------------------------|-------------------------------|-----------------------------|
| hm0   | 32.73                  | 14.74                         | 16.67                         | 0.11                        |
| prn0  | 22.21                  | 45.67                         | 19.34                         | 0.14                        |
| proj0 | 5.85                   | 35.68                         | 81.83                         | 0.01                        |
| mds0  | 30.65                  | 47.42                         | 14.48                         | 0.01                        |

The target SLC and MLC SSDs were configured as follows [9]. Both types of SSDs comprised four chips connected with two parallel channels. Each chip consisted of four dies, and each die had four planes. In SLC SSD, the size of a NAND page was 2 kB, and the size of a block was 128 kB. The NAND page read,
page write, and block erase latency were 25 µs, 200 µs, and 2 ms, respectively. In MLC SSD, the size of a NAND page was 4 kB, and the size of a block was 512 kB. The NAND page read, page write, and block erase latency were 500 µs, 900 µs, and 3.5 ms, respectively. The time needed to send one byte via the channel was 25 ns and the overprovision rate was set to 20%. Garbage collection was initiated when the ratio of clean blocks fell below 10%. Target chip, die, and plane of write requests were dynamically determined based on the current state of the channels and chips [8, 9], and the selective multiplane policy that maximises the internal parallelism of SSDs was used [9].

Fig. 2 is a graph of the average response time relative to the method that does not use shallow write, which is labelled as ‘base’. The label ‘host-shallow’ refers to the method that applies shallow write to all the host writes, and ‘selective-shallow’ indicates the proposed selective shallow write policy. In the ‘selective-shallow’ policy, the threshold for identifying short-lived data is fixed to 16 sectors. As is evident in the figure, the proposed policy reduces the average response time for all the traces. The reduction is more pronounced in the MLC SSD. The average response time is reduced by 8.50–41.14% in the MLC SSD, and by 1.25–24.85% in the SLC SSD. In the MLC SSD, the deep write is relatively slower than in the SLC SSD, and the parallelism effect is slightly less because of the large page size; thus, using shallow write is more beneficial. Meanwhile, host-shallow is inferior to the proposed policy. In fact, for host-shallow, some traces are much worse than the method that does not use shallow write—specifically for MLC SSD, by 36.54% (PROJ0) and by 125.57% (HM0). This is because the overhead of erroneously predicting the lifetime of the data and rewriting it in deep write is huge as all host write requests are processed in shallow write.

Meanwhile, the proposed policy has the overhead arising from rewriting the shallowly written data in deep write mode when the lifetime prediction is incorrect, which increases the total block erasure count. Thus, we present the increase ratio of the total block erasure count compared to the method that does not use shallow write in Fig. 3. The results show that the proposed policy increases the number of block erasures in most traces, with the increase ratio ranging from 0.37 to 5.12% in...
the SLC SSD and from 0.71 to 5.60% in the MLC SSD. The increase in the block erasure count is inevitable but not considerable, which indicates that the lifetime prediction using the request size is reasonable. In contrast, the host-shallow policy has a serious side effect as the total block erasure count increases from 2.76 to 33.80% in the SLC SSD and from 6.39 to 31.11% in the MLC SSD. Because the host-shallow policy regards all host writes as short-live data, the possibility of incorrect prediction is high. Consequently, the rewrite overhead is substantial, which results in increased block erasure count.

Meanwhile, the threshold for identifying short-lived data affects the performance of the selective shallow policy. Thus, to know the effect of the threshold, Fig. 4 depicts the relative average response time while varying the threshold from 8 sectors to 256 sectors. The relative average response time is calculated by regarding the result when the threshold value is fixed to 16 sectors as one. The result shows that if the threshold is 128 or more, the performance degrades severely and is similar to that of the host-shallow scheme because most write requests are evaluated as short-lived data. However, if the threshold is set to between 16 sectors and 64 sectors, the performance is not sensitive to the threshold. The performance difference ranges from −5.37 to 4.79% in all the traces, which implies that each threshold delivers a similar performance to the best case and that the threshold effectively identifies short-lived data.
5 Conclusion

To utilise a high percentage of short-lived data in server workloads, this paper proposed the selective shallow write policy that predicted the lifetime of data based on the write request size, and applied the shallow and fast write mode for short-lived data and the deep and slow write mode for long-lived data. Evaluation results showed that the proposed policy improved the average performance in all traces, with improvements of up to 41.14%. Further, the increase in the block erasure count caused by incorrect prediction and the rewriting overhead was limited to a maximum of 5.60%. In conclusion, for NAND flash memory supporting the two write modes was effective in improving SSD performance, and data lifetime prediction using write request size was effective.

The limitation of the work is that the proposed scheme will not improve the performance of read intensive servers because the shallow write reduces only the latency of a write operation. In addition, the shallow write reduces the margin for distinguishing each state of a NAND cell, and thus the read disturbance problem can be more serious. Therefore, as future work, we plan to apply the methods to mitigate the read disturbance problem together with the shallow write scheme. In addition, more accurate prediction method of data lifetime should be designed to reduce the re-writing overhead and the block erase count increase.

Acknowledgements

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