FastGC: Accelerate Garbage Collection via an Efficient Copyback-based Data Migration in SSDs

Fei Wu*, Jiaona Zhou*, Shunzhuo Wang*, Yajuan Du*‡, Chengmo Yang† and Changsheng Xie*
*Huazhong University of Science and Technology, China
‡City University of Hong Kong, China
† University of Delaware, USA
Corresponding author: Changsheng Xie, cs_xie@hust.edu.cn
{wufei, zhoujiaona, wangshunzhuo}@hust.edu.cn, dyjcityu2013@gmail.com, chengmo@udel.edu

ABSTRACT

Copyback is an advanced command contributing to accelerating data migration in garbage collection (GC). Unfortunately, detecting copyback feasibility (whether copyback can be carried out with assurable reliability) against data corruption in the traditional copyback-based GC causes an expensive performance penalty. This paper first explores copyback error characteristics on real NAND flash chips, then proposes a fast garbage collection scheme called FastGC. It utilizes copyback error characteristics to efficiently detect copyback feasibility of data instead of transferring out all valid data for detecting. Experiment results in the SSDsim show the proposed FastGC greatly promotes write response time and read response time by up to 44.2% and 66.3% respectively, compared to the traditional copyback-based GC.

1 INTRODUCTION

NAND flash-based Solid State Drives (SSDs) [1], given their fast access speed, low energy consumption, and high shock resistance, are widely used in portable devices, personal computers and enterprise storage systems as persistent storage devices [8]. Unlike HDD, NAND flash memory technology has the unique *out-of-place update* property, that is, a physical page is not allowed to be overwritten until an expensive erase operation is executed. Unfortunately, the basic unit of erase operation is not a page but a block which typically contains 32 to 1024 pages. Since erasing a block is quite costly and time-consuming, SSDs adopt a *remap-on-write* mechanism so as to avoid erasing the block upon every page updates. Pages with stale data, referred to as the *invalid pages*, can later be reclaimed and reused through garbage collection.

Garbage collection [5] is a significant operation in SSDs to reclaim invalid data attributed to the out-of-place update feature of NAND flash memory. GC is time-consuming and frequently triggered all over the lifetime of SSDs, which greatly influences system performance. Furthermore, valid page migration is a key process in GC and time overhead of data migration accounts for a large proportion in GC time overhead.

NAND flash manufacturers provide an advanced operation, called *copyback* or *Internal Data Move* [15], which moves data within one NAND flash plane. Copyback operation doesn't involve the SSD controller and requires no encoding or decoding, which can reduce data migration latency. Specifically, migrating page data of 16KB via copyback is 45.1% faster than traditional data migration. However, copyback operations can't be absolutely applied to speed up GC. Because error correction engine locates inside the SSD controller rather than inside each flash plane, data errors loaded from the source page cannot be detected/corrected during copyback operation. For data having been migrated for many times via copyback, their bit errors are accumulated and potentially exceed the error correction capability of Error Correction Code (ECC), causing data loss. To ensure that the data integrity is not impacted by accumulative bit errors caused by copyback, copyback feasibility detecting must be carried out prior to data migration.

Existing work has been proposed to employ copyback-based data migration in GC [3, 12, 15]. Unfortunately, these schemes bring about some extra performance penalties and hardware overhead. Moreover, they neglect error correction capability of ECC and copyback-based data migration are conservatively carried out, leading to limited promotion of GC performance.

To efficiently utilize copyback-based data migration in GC to promote system performance, this paper first characterizes the impacts of copyback on corrupting data under different Program/Erase(P/E) cycles. Experiments on real 3D TLC NAND flash chips show (1) in the conditions of assurable reliability, copyback is able to be carried out for a certain number of times under different P/E cycles and (2) the times gradually decease with P/E cycles increase, ending up at zero. Motivated by the observation, an efficient copyback feasibility detector is proposed, which utilizes copyback execution counts of data as a new indicator of copyback feasibility. Then this paper proposes a fast garbage collection scheme, called FastGC, which utilizes the new copyback-based data migration scheme to accelerate GC. FastGC employs the proposed efficient copyback feasibility detector to accelerate detecting whether data are migrated via copyback or not with slight performance overhead. Specifically, data undergoing a small number of copyback operations are migrated via copyback to accelerate its migration speed, while data reaching the threshold copyback counts are migrated via external data move to eliminate accumulated errors. The proposed scheme greatly improves garbage collection efficiency with assurable reliability. In a nutshell, this paper contributes to the state-of-the-art in the following aspects.

- To the best of our knowledge, this is the first work to explore copyback error characteristics and its reliability impacts on real flash chips.
- It proposes a new copyback feasibility detector based on copyback error characteristics.
- It proposes an accelerated garbage collection scheme which utilizes the new copyback feasibility detector to greatly improve garbage collection efficiency within assurable reliability.
- Model analysis and comprehensive simulation experiments are utilized to evaluate the proposed FastGC. The experiment results show that read and write response time are improved by up to 66.3% and 44.2%, respectively.

The rest of this article is organized as follows: Section 2 presents the related work. The motivations of this work are present in section 3. Section 4 introduces FastGC scheme and

DAC '18, June 24–29, 2018, San Francisco, CA, USA
© 2018 ACM. 978-1-4503-5700-5/18/06...$15.00
DOI: https://dx.doi.org/10.1145/3195970.3196051

analyzes the design of the proposed management strategy. Section 5 introduces experiments and summarizes the experiment results on the system performance improvement. The conclusion is given in section 6.

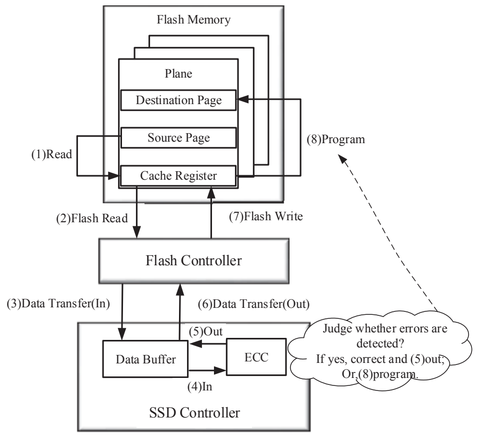


Figure 1: Traditional copyback-based data migration in GC. Data are first transferred to the SSD controller and then decoded by ECC to judge how to migrate data.

2 RELATED WORK

2.1 Garbage Collection

Garbage collection is a key operation to reclaim stale data for more free space. SSDs are usually over-provisioned with a certain amount of free blocks to support GC. A complete GC typically includes four steps: (1) selecting some blocks that contain some stale data as *victim blocks*, (2) copying valid data from the victim blocks to the free blocks, (3) modifying the mapping table, (4) erasing the victim blocks. When free blocks are going to be run out, GC is triggered and all valid data within the victim block need to be migrated. GC operation amplifies the number of read/write operations from host and can block I/O requests. During the whole lifetime of SSDs, GC is frequently triggered and greatly influences system performance, which brings about jitter performance of SSDs. GC tremendously degrades the worst-case response time of write requests by several orders of magnitude due to the excessive and unbounded overhead of valid page copying in garbage collection, as shown in Figure 2(a).

Many researches have been proposed to improve garbage collection efficiency. Some studies are aimed to reduce the frequency of GC by selecting victim blocks with cost-effective criteria [5, 13]. Some work focuses on how to identify and select victim blocks without extensively scanning flash memory for the status of data stored in the storage [17]. Some existing work exploits to separate different hotness data in different blocks, which reduces the GC overhead remarkably [4, 14]. Besides, some existing studies [3, 12, 15] aim to relieve the data migration overhead, while these schemes bring about the performance penalty or the hardware overhead. What is more, they don't take error correction capability of ECC into account and data migration promotion brought about by copyback is not remarkable.

2.2 Copyback-based Data Migration

External data move (EDM) is a reliable but time-consuming operation to migrate data from the source block to the destination block. Migrating the valid data from the source block to the destination block in EDM requires data to be first read to the cache register and then transferred data from the cache register to the SSD controller (steps (1)∼(3) in Figure 1). Then data are decoded and errors, if any, are corrected with ECC (steps (4)∼(5) in Figure 1). Data is finally re-encoded and written to the destination block in the NAND Flash array (steps (6)∼(8) in Figure 1).

Copyback, also called internal data move, is an advanced command supported by SSD manufacturers. Different from EDM, copyback makes it possible to transfer data within a plane from one page to another using the cache register in the plane (step (1) and step (8) in Figure 1). Besides, copyback moves data within the same plane without transferring data to the controller. It is considerably 45.1% faster than using EDM to migrate data [1]. While bit errors of data cannot be detected and corrected since data don't flow to the ECC module in the controller, which causes bit errors accumulated and increases the risk of data loss.

Traditional copyback-based GC (TCBGC for short) is a reliable but conservative method to reliably employ copyback [15]. Data are first migrated from the NAND flash array to the SSD controller (steps (1) ∼ (3) in Figure 1), then data are checked (step (4) in Figure 1). If any error exists, data are corrected and input to the NAND flash array (steps (5) ∼ (8) in Figure 1). Otherwise, data are migrated via copyback (step (8) in Figure 1). Chang et al. [3] add a hardware logic in flash controller to detect/correct bit errors of data. Bit errors are able to be detected and corrected in the flash controller instead of the SSD controller, reducing the data transmission delay. The technique is helpful to reduce performance penalty, but this brings extra hardware overhead. Jenong et al. [12] utilize shared I/O data path to migrate data from the source device to the destination device in the same channel to extend copyback and only return corrected bits rather than all data to NAND flash array (steps (6)(7) in Figure 1), which decreases the size of transferred data. But that paper doesn't mention how to correctly combine the original data and corrected bits in flash array and it may require extra hardware overhead.

3 MOTIVATION

3.1 Heavy Data Transmission Overhead

Garbage collection is a key but time-consuming operation in SSDs and causes system performance fluctuation, as shown in Figure 2(a). The whole time overhead of GC T_{GC} is made up of two parts, including data migration time T_{move} and erase latency T_{erase}. Data are first read from the source page to the cache register in the plane then data are transferred to the data buffer in SSD controller. After detecting/correcting bit errors, data are transferred back to the cache register and written into the destination page. The time overhead of one page data migration is

$$T_{movepg} = T_{read} + 2 \times T_{trans} + T_{program},$$

where T_{read}, T_{trans} and $T_{program}$ represent the latency of one flash read, one page data transmission and one flash write, respectively. The proportion of data transmission latency L_{trans} in GC time overhead is

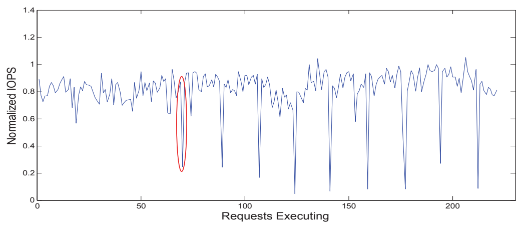
$$\frac{L_{trans}}{T_{GC}} = \frac{2\mu N_{page} \times T_{trans}}{\mu N_{page} \times (T_{read} + 2 \times T_{trans} + T_{program}) + T_{erase}},$$

where N_{page} presents total number of pages in one block and μ is the ratio of valid to total pages in a victim block. μ is related to request size and over-provisioned space in SSD. Figure 2(b) shows the proportion of data transmission latency on GC time overhead. As migrated pages increase, data transmission latency becomes vast and greatly influences GC performance.

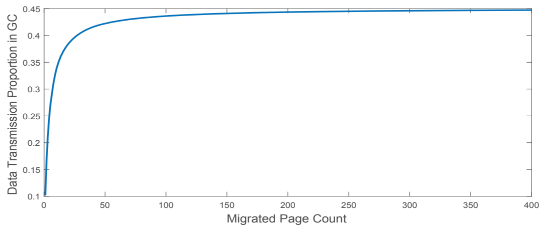
3.2 Shortcomings of TCBGC

The traditional copyback-based GC efficiently avoids bit error accumulation, but *its copyback feasibility detection overhead is expensive and can't make full use of copyback to promote GC performance.* On the one hand, the overhead of detecting copyback feasibility is high and consumes the performance promotion

[1]The value is calculated according to datasheet [16].



(a) GC influence on system performance. This shows how IOPS varies with requests carried out and the curve valley (in red ellipse) comes out once garbage collection is triggered. The experiment data is from simulation with Financial 1 trace [7].



(b) Proportion of data transmission latency in GC time overhead. It shows how proportion of data transmission varies with migrated page count in one block increases. The data are calculated according to the NAND flash characteristics[16], as shown in Table 1.

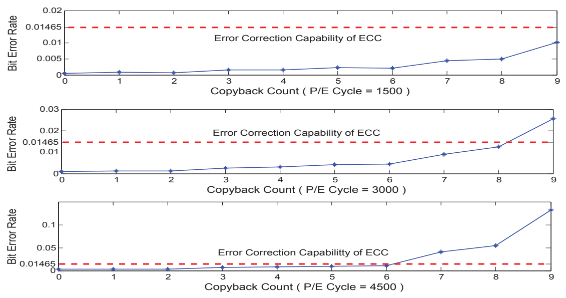
Figure 2: The influence of data transmission on system performance.

brought about by copyback. Data need to be read to ECC modules in the SSD controller, then data are decoded for detecting bit errors. This causes extra transmission latency and decode latency, which are a great penalty. On the other hand, the copyback feasibility detection in traditional copyback-based GC is conservative. This scheme neglects error correction capability of ECC and copyback is rarely carried out due to high bit errors of 3D TLC NAND flash memory, as shown in Figure 3(a).

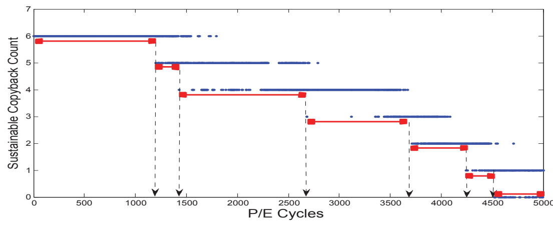
3.3 Copyback Error Characteristics

As we know, ECC is an efficient policy to help increase the reliability of SSDs. And it is able to correct a certain number of errors in flash pages. Although there is no chance to perform error correction during the copyback operation, ECC is capable of protecting against data loss within a certain number of copyback operations. Experiments based on the real 3D TLC NAND flash chips [16] are carried out to illustrate copyback error characteristics in this section.

SSDs are complex electronic storage medium prone to failure mechanisms mainly related to their basic component: NAND flash memory. For some applications with high access frequency, data bit errors caused by retention is slight [2]. To characterize copyback errors in practical high access frequency applications, this paper takes program interference, read disturb and P/E cycles [2] into experiments. ECC with 120 bits correction per 1024 bytes as shown in the datasheet [16] is adopted in the experiments. A block size of random data is generated as base data, and every data page is randomly migrated into one flash page of another free block with the same P/E cycle via copyback operation. These data are migrated among the blocks with the same P/E cycle in this way as long as ECC is able to correct bit errors of data. As copyback is carried out, bit errors of data gradually rise. Furthermore, the upward trend of bit errors grows with P/E cycle increasing, as shown in Figure 3(a). Recorded maximal copyback counts with certain P/E cycles are called *threshold copyback counts*, as shown in Figure 3(b). Figure 3(b) shows threshold copyback counts with different P/E cycles and threshold copyback counts have declined as P/E cycle increases. The secure threshold copyback counts with different P/E cycles



(a) The relationship between copyback counts and raw bit error rates.



(b) Sustainable copyback count with different P/E cycles. Every blue dot presents threshold copyback count with a certain P/E cycle and red line shows secure threshold copyback count with different P/E cycles.

Figure 3: Copyback error characteristics.

(red line in the Figure 3(b)) are applied to copyback feasibility detection to relieve the performance penalty.

Motivated by the observation of copyback error characteristics, an efficient copyback feasibility detector which utilizes the threshold copyback count as an indicator to accelerate copyback feasibility detection.

4 FASTGC

A fast garbage collection, called FastGC, is proposed to accelerate GC via an efficient copyback-based data migration, as shown in Figure 4. FastGC adds a copyback feasibility detector module in the SSD controller to efficiently distinguish whether data can be migrated by copyback or not with a slight overhead. Friendly data organization is applied in FastGC, which is helpful to detect copyback feasibility. FastGC applies a copyback error aware data migration to move data. This scheme greatly reduces performance penalty for copyback feasibility detection and accelerates garbage collection.

4.1 FastGC Friendly Data Organization

This paper first utilizes copyback execution counts as the indicator to detect copyback feasibility. Thus copyback execution counts of all data must be recorded. Considering that block is the basic unit of GC, this paper takes the block as the management unit of data copyback execution counts. So this paper divides every block into data space and metadata space, as shown in Figure 5. Data space is for user data and their corresponding copyback execution counts are stored in metadata space. Metadata space is located in the last one or more pages of a block. The data organization not only avoids unnecessary RAM overhead to maintain copyback metadata but also is helpful to quickly find its copyback execution counts for all user data.

The space overhead of metadata is acceptable. User data are usually written and migrated in form of page, referred to *LPN* (logical page number). Copyback metadata of all user data is recorded in form of "LPN + copyback count". LPN indicates user data and copyback count represents the corresponding copyback execution count. The threshold copyback counts are all less than 10 with different P/E cycles according to the tested copyback

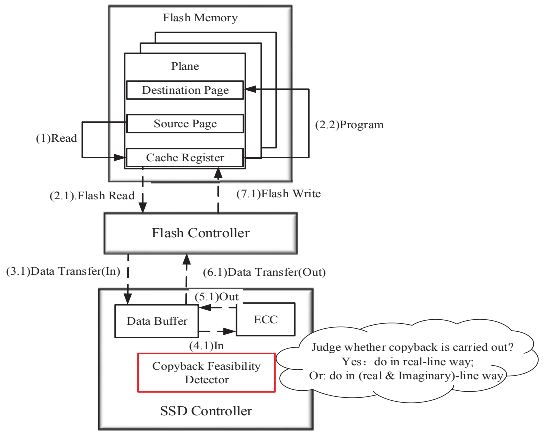


Figure 4: Efficient copyback-based data migration in FastGC. The migration way of all valid data in the victim block is decided in advance by additional copyback feasibility detector in the SSD controller.

error characteristics. As a result, for every copyback metadata entry, a LPN (4B space overhead) and copyback count (1B space overhead) need to be maintained. It takes only 5B storage space in total for one entry and maintaining all copyback metadata needs acceptable storage space. For example, for 1TB SSD, its flash page is 16KB and one block includes 576 pages. The last one page of every block is enough to store all copyback metadata. There is $1TB/(576 \times 16KB) \approx 116508$ blocks in the SSD and the total size of copyback metadata in 1TB SSD is $116508 \times 16KB \approx 1.6GB$.

4.2 Copyback Feasibility Detector

An Efficient copyback feasibility detector based on the copyback error characteristics is elaborated in this section. The copyback feasibility detector utilizes threshold copyback counts and copyback execution counts of all data to decide which way of migration for all data. Threshold copyback counts are maintained in RAM for high access speed. The data migration way is decided by simply comparing copyback execution counts with corresponding threshold copyback counts.

For a victim block to be reclaimed, copyback metadata of the victim block are first read to the SSD controller before user data are migrated. Copyback feasibility of all data are detected by comparing the copyback execution counts of data with the threshold copyback count of the destination free block. If the copyback execution counts of the data are smaller than the threshold copyback count of the free block, it means the data can still be migrated by copyback, but the opposite is not. By reading copyback metadata instead of all data and some simple comparisions, copyback feasibility of all data is efficiently detected, which greatly reduces the performance penalty caused by the detection of copyback feasibility .

4.3 Copyback Error Aware Data Migration

When garbage collection is triggered, the migration way of all data in the victim block is decided by the copyback feasibility detector prior to data migration. FastGC utilizes a copyback error aware data migration to promote GC performance. Valid data with small copyback counts are migrated via copyback, external data move is carried out once the copyback counts exceeds the threshold copyback counts of free blocks. Only one external data move can relieve accumulated bit errors after executing multiple copyback operations and then copyback is adopted to accelerate data migration. Some extra read and write processes need to be carried out to maintain and obtain

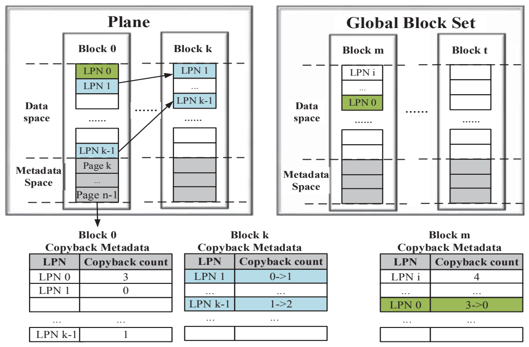


Figure 5: Data organization in FastGC.

copyback metadata in the copyback feasibility detector, but the overhead is much smaller than transfer latency and decoding latency caused by detecting bit errors since the number of data migration is large. Thus, FastGC is able to efficiently promote GC performance. At the end of lifetime of SSDs, blocks have undergone a lot of P/E cycles and they are error-prone via copyback. So metadata space is merged to data space and all data are migrated via external data move for data integrity.

For newly written data, FastGC adds and maintains a copyback metadata entry in RAM. The copyback count of the entry is set to 0. Once data space of a free block is run out, copyback metadata of the free block are flushed to corresponding metadata space of the free block. When GC is triggered, the SSD controller applies for two free blocks. One block is for copyback in the same plane as the victim block and the other block derives from the global block set for external data move. Copyback metadata are read to the copyback feasibility detector to detect their copyback feasibility. For data being unable to be migrated via copyback, they are read to the SSD controller (steps (1), (2.1), (3.1) in Figure 4) and written to flash memory (steps (5.1), (6.1), (7.1), (2.2) in Figure 4) again after correcting bit errors (steps (4.1), (5.1) in Figure 4). For example, garbage collection is triggered and block 0 is selected as the victim block, as shown in Figure 5. Block k and block m are the selected free blocks, block k is in the same plane as block 0 and block m is in global block set. The SSD controller obtains that the threshold copyback count of block k is 3 according to its P/E cycle, then compares the threshold copyback count of block k with the copyback counts of valid data in block 0, respectively. The copyback count of LPN 0 is more than the threshold copyback count of block k, so it is migrated to block m and a new copyback entry with copyback count 0 is added, as shown green entries in Figure 5. LPN 1 and LPN k-1 are able to be migrated via copyback, so they are migrated to block k and their copyback counts plus 1, as shown blue entries in Figure 5.

5 EVALUATION

5.1 Analysis of FastGC

In this section, this paper derives the analysis model to evaluate performance promotion and overhead of FastGC. Write amplification is a key metric to evaluate system overhead and system response time for reads and writes play a significant role to evaluate performance. This paper uses these metrics to assess proposed FastGC.

To this end, assume that all data are completely random accessed and ideal wear-leveling scheme is employed in the SSD. That is, there is the same probability for all data to be read and written and to be allocated in all blocks. When the SSD is in the saturated state, all data are evenly distributed in the SSD. At this point,

$$\mu = \frac{V_{user}}{V_{SSD}} = \frac{1}{1+OP\%},$$

where V_{user} represents user visible space, V_{SSD} is the actual capability and $OP\%$ is the ratio of over-provisioning space to the total space of the SSD. SSD manufactures usually provide $7\% \sim 28\%$ over-provisioning space, thus μ is $0.78125 \sim 0.935$.

5.1.1 Write Amplification. The Write Amplification Factor (WAF) [9] is defined as follows:

$$WAF(\mu) = \frac{\mu \times N_{page}}{(1-\mu) \times N_{page}} = \frac{\mu}{1-\mu}. \quad (1)$$

It is the ratio of migrated valid data to reclaimed invalid data. Assuming that total write request amount is V_{w_req}, S_{page} presents the size of a page, the total write amount is

$$\begin{aligned} V_w &= \frac{V_{w\_req}}{S_{page} \times N_{page}} \times WAF(\mu) \times S_{page} \\ &= \frac{\mu V_{w\_req}}{(1-\mu) N_{page}}. \end{aligned}$$

Because metadata space is much smaller than SSD size (the ratio is N_{meta}/N_{page}, where N_{meta} presents the number of metadata pages in a block.), the ratio of valid page migration in FastGC is approximate to μ. Metadata space is specialized for copyback metadata rather than user data, and copyback metadata are updated and written to the new block during GC process. So WAF of FastGC is

$$WAF_{FastGC}(\mu) = \frac{\mu \times N_{page} + N_{meta}}{(1-\mu) \times N_{page}}. \quad (2)$$

And the total write amount of FastGC is

$$\begin{aligned} V_{FastGC\_w} &= \frac{V_{w\_req} \times S_{page}}{S_{page} \times (N_{page} - N_{meta})} \times WAF_{FastGC}(\mu) \\ &= \frac{\mu \times N_{page} + N_{meta}}{\mu \times (N_{page} - N_{meta})} \times V_w. \end{aligned}$$

Because meta page count (N_{meta}) is much smaller than page count per one block (N_{page}) and migrated page count ($u \cdot N_{page}$), $\frac{\mu \times N_{page} + N_{meta}}{\mu \times (N_{page} - N_{meta})} \to 1$ and $V_w \approx V_{FastGC\_w}$, write amplification of FastGC hardly gets worse.

5.1.2 Performance Analysis. Traditional copyback-based GC does some extra reads to detect bit errors for copyback feasibility, and it causes inevitable performance penalty. However, FastGC detects whether copyback is carried out or not with less reads and some simple comparisons when GC is triggered. In this section, data migration performance promotion is illustrated during one GC process.

Assuming that T_t, T_r and T_p present the time of transferring one page, reading one page and writing one page, respectively. The number of valid page in FastGC is $N_m = \mu N_{page}$ and the latency of migrating one page is $T_m = T_r + 2T_t + T_p$, the overhead of migrating valid data in the victim block via external data move is $O_{EDM} = N_m T_m$. The overhead of data migration in FastGC is

$$\begin{aligned} O_{FastGC}(N_m) &= N_{meta} \times (T_r + 2T_t + T_p) + N_m \times (T_r + T_w) \\ &= O_{EDM} + (N_{meta} T_m - 2 N_m T_t). \end{aligned}$$

Thus the performance promotion of FastGC is $N_{meta} T_m - 2 N_m T_t$. The valid pages that are migrated via copyback are more, data migration is faster completed and performance promotion per one GC is greater.

5.2 Experiment Setup

Simulations based on SSDsim [10] were implemented to evaluate the proposed FastGC, compared to the GC scheme with the traditional copyback-based data migration. The simulator parameters are presented in Table 1. In the experiments, 20% of flash memory space is preserved as over-provisioning space. FastGC can apply to the most address mapping schemes and works together with other GC schemes discussed above. For simplicity, all simulations uniformly adopt page mapping scheme and greedy algorithm for selecting victim blocks. Five real-world workloads are selected for simulations. Financial 1 and Financial 2 [7] are from OLTP applications running at two large financial institutions. Hm0 and Exchange [11] are 1-week block I/O traces collected from enterprise servers at Microsoft Research Cambridge. CloudVPS [6] is collected from the Cloud VPS public cloud system. The I/O characteristics of typical workloads are shown in Table 2.

Table 1: NAND FLASH CHARACTERISTICS

page size	16KB	block size	9MB (576 pages)
read time	91µs	program time one shot	706 µs
erase time	5ms	bus timne(x8)	20 ns
copyback time	2209µs	ECC ability	120 bits per 1KB

Table 2: I/O characteristics of 5 typical workloads

	# of request	Write ratio	# of migrated page per GC	Ratio of GC write
Financial 1	5334944	76.83%	∼406	70.6%
Financial 2	3678863	17.75%	∼370	64.2%
Exchange	599272	67.96%	∼158	27.5%
hm0	3993315	64.50%	∼74	12.8%
VPS	6451271	51.04%	∼347	60.33%

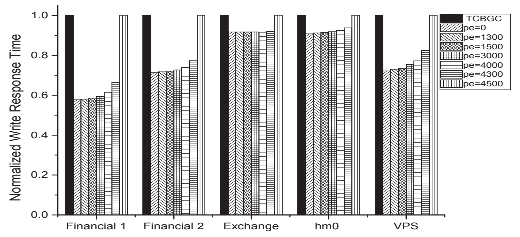
To comprehensively evaluate the influence of FastGC on the system performance, simulations in 7 different stages of lifetime of SSDs are carried out. Initial P/E cycles are set to 0, 1300, 1500, 3000, 4000, 4300 and 4500, respectively. Accordingly, threshold copyback counts of SSDs are from 6 to 0 according to copyback error characteristics, as shown in Figure 3(b).

5.3 Experiment Results

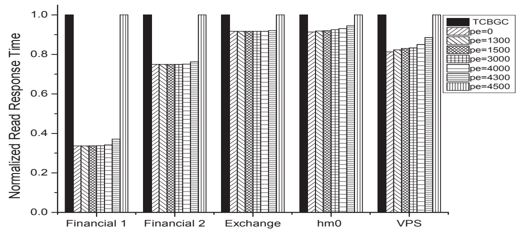
5.3.1 Performance Promotion. Figure 6(a) and Figure 6(b) show response time in write and read, respectively. Experiments present that FastGC is effective to improve system performance. When SSDs are in early lifetime, threshold copyback count is relatively large and the performance promotion is obvious. In the later stage of lifetime, although threshold copyback count is small, the performance promotion is still obvious by utilizing FastGC. At the end of lifetime (P/E cycles are beyond 4500), FastGC just adopts external data move to ensure reliability. There is no need to store copyback metadata. System performance is equal to that of traditional copyback-based GC.

When initialized P/E cycle is 0 and threshold copyback count is 6, write response time promotions are 42.2%, 28.5%, 8.3%, 9.2% and 17.8% under Financial 1, Financial 2, Exchange, hm0 and VPS, respectively. Read response time promotions are 66.3%, 25.0%, 8.3%, 8.6% and 18.7% under the five real-world workloads above, respectively. When threshold copyback count is 1, write response time promotions are 33.5%, 22.8%, 8.1%, 6.3% and 17.6%, respectively. Read response time promotions are 62.9%, 23.7%, 7.9%, 5.5% and 11.4% under the five workloads, respectively. Threshold copyback count is more, the number of data migration via copyback is more. Thus GC efficacy is higher. The reasons why the performance promotion under Financial 1 is the most significant are that average migrated pages in a GC process are great and pages migrated via copyback account for large proportion in all flash write operations. Exchange includes hot data (Their access frequency is high) and the maximal copyback count of data is no more than 2, so the system performance is unchanged when P/E cycles vary from 4300.

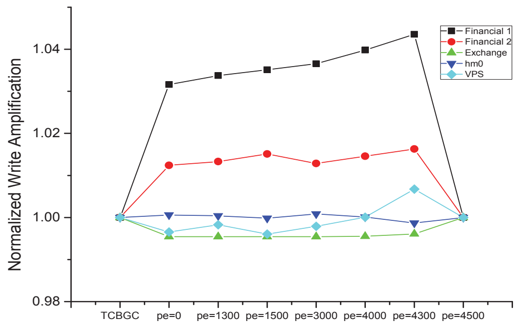
5.3.2 Write amplification. Write amplification plays a significant role in measuring system overhead of proposed FastGC and traditional copyback-based GC. Figure 6(c) presents the experiment results of write amplification, and the results show that write amplification under five workloads hardly gets worse,



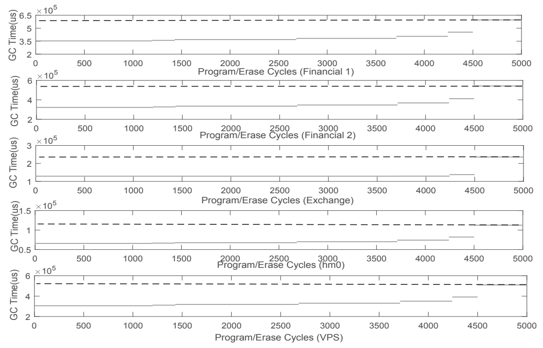
(a) Write Response time. TCBGC is short for traditional copyback-based GC. "pe=N" represents the initial P/E cycle is N.



(b) Read Response time. The legends are the same as (a).



(c) Write Amplification.



(d) Fast data transmission in FastGC. Imaginary lines present the time overhead of traditional copyback-based GC.

Figure 6: Simulation results.

just like the analysis mentioned above. The write amplifications under five traces are 104.4%, 101.6%, 99.6%, 99.9% and 100.7%, respectively. In experiments, the copyback metadata in every block only takes one page. The metadata space is much smaller than capability of SSDs, and extra writes are positively correlated with space wastes. Thus write amplification is essentially unchanged. Not only the space size but also the request access can influence write amplification, thus write amplifications under Exchange and hm0 slightly reduce.

5.3.3 GC Performance Promotion. Figure 6(d) shows the GC performance improvement brought about by FastGC, compared to the traditional copyback-based GC. Experiments verify the performance advantage of FastGC in section 5.1.2. For the five typical traces, FastGC efficiently decreases the time overhead of GC. In details, the more data are migrated, the more performance improvement FastGC brings about. Furthermore, GC efficacy gradually degrades as threshold copyback counts decrease.

6 CONCLUSION

This paper focuses on reducing the expensive performance penalty of detecting copyback feasibility in traditional copyback-based GC. Motivated by the copyback error characteristics tested on the real flash chips, FastGC is proposed to relieve these issues. An efficient copyback feasibility utilizes copyback execution counts as the indicator to determine the way of data migration with slight performance overhead and acceptable space overhead. Comprehensive experiments show that FastGC greatly upgrades system performance.

ACKNOWLEDGMENTS

This work is sponsored by the National Natural Science Foundation of China under Grant No. U1709220No.61472152Wuhan Applied Basic Research Project No. 2017010201010108 Shenzhen Basic Research Project No. ICYJ20170307160135308, the Fundamental Research Funds for the Central Universities No.2016YXMS019 and the 111 Project (No. B07038). This work is also supported by Key Laboratory of Data Storage System, Ministry of Education.

REFERENCES

[1] N. Agrawal, V. Prabhakaran, and et al. 2008. Design Tradeoffs for SSD Performance. *Proc. of Usenix Technical Conference* (2008), 57–70.
[2] Y. Cai, E. Haratsch, and et al. 2012. Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis. *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2012), 521–526.
[3] W. Chang, Y. Lim, and J. Cho. 2014. An Efficient Copy-Back Operation Scheme Using Dedicated Flash Memory Controller in Solid-State Disks. *Proc. of the International Journal of Electrical Energy* 2 (2014), 1.
[4] M. Chiang, P Lee, and R. Chang. 1999. Using Data Clustering to Improve Cleaning Performance for Flash Memory. *Proc. of the Software-Practices and Experiences* 29(3) (1999), 267–290.
[5] M. L. Chiang and R. C. Chang. 1999. Cleaning Policies in Mobile Computers Using Flash Momory. *Proc. of The Journal of Systems and Software* 48 (1999), 213–231.
[6] CloudVPS. 2013. VPS Trace. *https://www.cloudvps.nl/* (2013).
[7] Laboratory for Advanced System Software. 2007. UMass Trace Repository. *http://traces.cs.umass.edu/index.php/Storage/Storage* (2007).
[8] J. Gray and B. Fitzgerald. 2008. Flash disk opportunity for server applications. *Proc. of Queue* 6(4) (2008), 1823.
[9] X. Hu, E. Eleftheriou, and et al. 2009. Write amplification analysis in flash-based solid state drives. *The Israeli Experimental Systems Conference* (2009), 10:110:9.
[10] Y. Hu, H. Jang, and et al. 2012. Exploring and Exploiting the Multilevel Parallelism Inside SSDs for Improved Performance and Endurance. *Proc. of TOC* 62 (2012), 1141–1151.
[11] SNIA IOTTA. 2007. MSR Cambridge Traces. *http://iotta.snia.org/traces/388* (2007).
[12] J. Jeong and Y. Song. 2012. A Technique to Improve Garbage Collection Performance for NAND Flash-based Storage Systems. *Proc. of the IEEE Transactions on Consumer Electronics* 58(2) (2012), 470–478.
[13] A. Kawaguchi, S. Nishioka, and H. Motoda. 1995. A Flash-Memory Based File System. *Proc. of the USENIX* (1995), 155–164.
[14] H. Kim and S. Lee. 1999. A New Flash Memory Management for Flash Storage System. *Proc. of the International Computer Software and Applications Conference* (1999), 284–289.
[15] Micron. 2006. TN-29-15: NAND Flash Internal Data Move Introduction. (2006).
[16] TOSHIBA. 2016. TOSHIBA 3D Flash Memory Toggle DDR2.0 Technical Data Sheet. (2016).
[17] C. Tsao, Y. Chang, and M. Chang. 2013. Performance Enhancement of Garbage Collection for Flash Storage Devices: An Efficient Victim Block Selection Design. *Proc. of the Design Automation Conference* 50 (2013), 165.