2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS) | 978-1-6654-4066-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/IPDPS49936.2021.00025

CAGC: A Content-aware Garbage Collection Scheme for Ultra-Low Latency Flash-based SSDs

Suzhen Wu‡, Chunfeng Du‡, Haijun Li‡, Hong Jiang*, Zhirong Shen‡, Bo Mao‡✉
‡School of Informatics at Xiamen University, Xiamen, Fujian, China
*Department of Computer Science & Engineering at University of Texas-Arlington, Texas, USA
✉Corresponding author: Bo Mao (maobo@xmu.edu.cn)

***Abstract*—With the advent of new flash-based memory technologies with ultra-low latency, directly applying inline data deduplication in flash-based storage devices can degrade the system performance since key deduplication operations lie on the shortened critical write path of such devices. To address the problem, we propose a Content-Aware Garbage Collection scheme (CAGC), which embeds the data deduplication into the data movement workflow of the Garbage Collection (GC) process in ultra-low latency flash-based SSDs. By parallelizing the operations of valid data pages migration, hash computing and flash block erase, the deduplication-induced performance overhead is alleviated and redundant page writes during the GC period are eliminated. To further reduce data writes and write amplification during GC, CAGC separates and stores data pages in different regions based on their reference counts. The performance evaluation of our CAGC prototype implemented in FlashSim shows that CAGC significantly reduces the number of flash blocks erased and data pages migrated during GC, leading to improved user I/O performance and reliability of ultra-low latency flash-based SSDs.**

***Index Terms*—Ultra-Low Latency Flash-based SSDs, Garbage Collection, Data Deduplication, Reference Count, Data Placement**

I. Introduction

Flash memory technology is disrupting the storage media market, leading to a significant evolutionary investment and innovation in the storage systems market [7]. Flash-based Solid State Disks (SSDs) have emerged as attractive alternatives to Hard Disk Drives (HDDs), increasingly replacing or coexisting with HDDs in smartphones, personal laptops, enterprise storage systems and large-scale data centers [14], [28]. However, due to the unique features of flash memory, such as asymmetric read-write performance, limited erase cycles, and garbage collection, data writing has an important impact on the performance and reliability of flash-based SSDs [1], [27], [30], [31]. Due to its ability to identify and reduce the writing of redundant data pages, the inline data deduplication technology can obviously improve the storage efficiency and reliability of flash-based memory systems, attracting extensive attention from both academia and industry [2], [11]. While inline data deduplication effectively reduces the amount of redundant write data to a flash storage device, it also increases the write response times due to the additional and expensive overhead of fingerprint calculations and lookup operations on the critical write path [37].

In order to address the aforementioned problems, researchers have proposed to use hardware coprocessors and sampling techniques to reduce the latency overhead caused by hash fingerprint calculations. For example, both CA-SSD [11] and paper [18] use on-chip hash calculation coprocessors to speed up the hash fingerprint calculation. CA-FTL [2] uses sampling and pre-hash techniques to reduce the number of data blocks that need expensive fingerprint calculation, thus reducing the latency overhead of fingerprint processing. However, the hash-coprocessor schemes introduce extra hardware overhead, and the sampling technique also needs to perform the hash fingerprint calculation on other data blocks. None of them can eliminate the latency overhead caused by the fingerprint calculation and lookup operations along the write I/O path.

With the rapid development and application of new flash storage technologies, such as Z-NAND and XL-Flash [4], [42], the performance of solid-state disks based on these flash medias has been improved so vastly that they are now referred to as ultra-low latency flash-based SSDs [13], [21]. However, directly applying the inline data deduplication technology to ultra-low latency flash-based SSDs will notably increase the response latency of user requests because the ultra-low latency on the critical path makes the deduplication-induced latency overhead much more pronounced, thus reducing the performance of deduplication-based flash storage devices. Preliminary experimental results show that a direct application of the inline data deduplication technology on Samsung Z-NAND SSDs increases the response latency by up to 71.9%, with an average increase of 43.1%.

In addition to the user's read and write requests, the flash-based SSDs also need to perform GC operations to reclaim the invalid data pages, and perform block erase operations to free up space for subsequent new user write data. Generally speaking, the GC procedure includes selecting the victim flash block, migrating the valid data pages in the victim flash block to other free flash blocks, erasing the victim flash block and marking it as free. The basic unit of erase operation is a flash block consisting multiple (hundreds of) pages, while the basic unit of a read and write request is a page. The latency of a block erase operation is an order of magnitude higher than that of a read or write request [1]. Thus, GC operations are very time-consuming background tasks inside flash-based SSDs that directly affect the foreground user read and write requests,

978-1-6654-4066-0/21/$31.00 ©2021 IEEE
DOI 10.1109/IPDPS49936.2021.00025

which significantly increases the user request response times and causes serious performance variability.

In view of the above challenges facing the direct application of the data deduplication technology in ultra-low latency flash-based SSDs, this paper proposes a two-pronged approach called Content-Aware Garbage Collection scheme (CAGC). On one hand, CAGC embeds the data deduplication technology into the GC process, which not only hides the overhead of fingerprint calculation, but also eliminates the write operations of redundant data pages. On the other hand, by exploiting the reference count feature of the data deduplication technology and grouping flash blocks into hot and cold regions, data pages with similar reference counts (larger than a threshold) are stored together in flash blocks of the cold region, and data pages with reference count of exactly 1 are stored together in flash blocks of the hot region. The overhead of moving data pages with high reference counts during GC is reduced because these data pages with high reference counts are usually much less likely to become invalid (as elaborated in Section III-C), which further improves the GC efficiency of ultra-low latency flash-based SSDs. The performance results show that the CAGC scheme significantly reduces the number of flash blocks erased and data pages migrated during GC, thus improving the performance and reliability of ultra-low latency flash-based SSDs. This paper makes the following contributions:

(1) From our preliminary experiments, we find that inline data deduplication significantly degrades the performance of ultra-low latency flash-based SSDs.
(2) To address the above challenge when directly applying data deduplication for ultra-low latency SSDs, we propose a content-aware GC scheme by exploiting the features of both GC and data deduplication.
(3) We conduct extensive experiments on a lightweight CAGC prototype and the evaluation results show that CAGC significantly improves the GC efficiency for ultra-low latency SSDs.

The rest of this paper is organized as follows. Background and motivation are presented in Section II. We describe the design details of the Content-Aware Garbage Collection scheme in Section III. The performance evaluation is presented in Section IV. The related work is presented in Section V. We conclude this paper in Section VI.

II. Background and Motivation

In this section, we first describe the development of ultra-low latency flash-based SSDs. Then we elaborate on why inline data deduplication is not suitable for ultra-low latency flash-based SSDs. Finally, we present the GC workflow in flash-based SSDs to motivate our new Content-Aware Garbage Collection optimization for ultra-low latency flash-based SSDs.

A. Ultra-low latency flash-based SSDs

With the advent of Samsung's Z-NAND and Toshiba's XL-Flash technologies, the I/O latency of Ultra-Low Latency (ULL) SSDs can be up to 10 times shorter than that of a high-performance NVMe SSD [13], [21], [42]. For example, Samsung's Z-NAND completes a 4KB-sized read service within 3us [4] while a general high-performance NVMe SSD completes an I/O service within 47-52us, including data transfer and FTL execution latencies [42]. In addition, other flash memory vendors also develop and promote low-latency flash memory chips, such as Toshiba's XL-Flash technology [42].

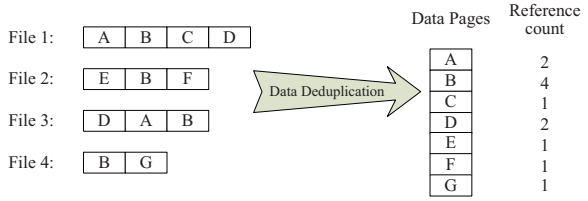


Fig. 1. An example of how data deduplication works.

Although ultra-low latency SSDs have low read and write latency, they are very sensitive to the amount of write data due to the asymmetric read and write performance and the limited endurance (erase cycles) of flash memory. The amount of data written to the flash memory directly affects the performance and reliability of flash-based SSDs. Therefore, reducing the amount of write data helps improve the performance and reliability of flash-based SSDs and extend their life.

B. Inline data deduplication for flash storage

Inline data deduplication means that data deduplication process is performed on the user write path before the data is written to the storage devices. Data deduplication is a specific type of data compression. It splits files into multiple data chunks that are uniquely identified by a fingerprint (i.e., a hash signature) of each individual data chunk. The redundant data chunks in a file are replaced by pointers to their stored unique copies. Figure 1 shows an example of how data deduplication works. The data deduplication technology has been demonstrated to be very effective in shortening the backup window and saving the network bandwidth and storage space in cloud backup, archiving and primary storage systems (e.g. flash-based storage systems).

Recent studies have shown that the ability of data deduplication to reduce the write traffic can significantly improve the performance and reliability of the flash storage systems. In fact, inline data deduplication has become a commodity feature in flash-based storage products for many leading companies, such as HPE Nimble Storage [34] and Pure Storage [5], [6], for the purpose of enhancing system performance, reliability and space efficiency.

However, despite of its great benefits, data deduplication has two important drawbacks, namely, expensive calculation and memory overheads on the critical I/O path, which adversely affects the performance of systems, especially for ultra-low latency flash-based SSDs. With the Z-NAND and XL-Flash technologies, the I/O latency of next-generation flash storage is significantly lower than earlier generations. On the other hand,

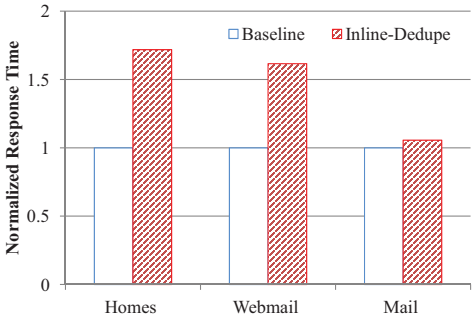


Fig. 2. The performance results of an ultra-low latency flash-based SSD with (Inline-Dedup) and without (Baseline) inline data deduplication, normalized to Baseline.

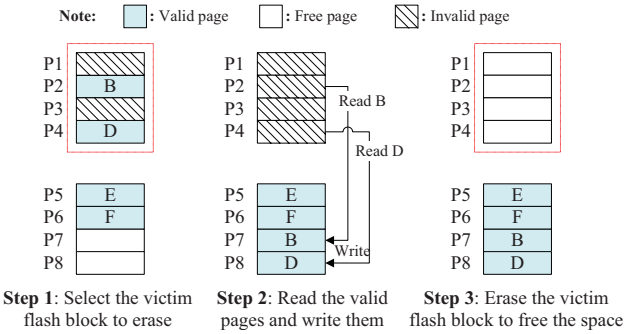


Fig. 3. 3 steps of a typical garbage collection process in SSDs.

the hash computing latencies of SHA-1/256 in deduplication-based systems remain largely unchanged, especially for the limited computing capability in flash-based SSDs. As a result, the performance bottleneck has been shifted from the flash storage to the deduplication process in such ultra-low latency flash storage systems [37].

Figure 2 shows the normalized performance results of an ultra-low latency flash-based SSD with and without inline data deduplication, driven by three FIU workloads [9]. We can see that inline data deduplication degrades the system performance for all the three workloads, even for the Mail workload with a 93% deduplication ratio. The reason is that all the incoming data blocks must go through the expensive hash computing and search processes whose latency is much larger than the I/O latency of the ultra-low latency flash storage.

C. Garbage collection and motivation

Due to the unique physical features of NAND flash, write requests are serviced out-of-place rather than in-place. In flash-based SSDs, data can only be written to erased pages (*a.k.a.*, free pages), where the in-place (before-write) pages become invalid (stale) after out-of-place write operations. After that, the invalid pages in a block, called a victim flash block, must be freed by copying (reads followed by writes) the data of the valid pages in the victim block into a free block before the victim block is erased, which makes free block available for subsequent write data. This process is known as garbage collection process that significantly affects the user I/O performance of SSD-based storage systems [15], [23], [39].

Reclaiming space used by invalid data without losing valid data is a major performance bottleneck in GC for flash storage devices. Thus, a typical victim flash block selection algorithm usually chooses the flash blocks that contain the maximum invalid data pages while simultaneously considering the block erase count and age information [33], [17], [10]. There are three main approaches that existing victim-block selection algorithms taking for garbage collection in flash-based SSDs. The first one, referred to as *random approach*, randomly selects the victim blocks with invalid pages for erasing, for ease of wear leveling and low selection overhead [29]. The second one selects the victim blocks with the most invalid pages, thus called *greedy approach* [10]. However, since the cold data pages of the CAGC system are stored together and the cold data pages are less likely to be invalidated, the use of greedy algorithms may lead to an uneven wear-leveling problem. To address this problem, a third approach, *cost-benefit approach*, is proposed to comprehensively consider both the number of invalid pages in the victim flash block and the erasing history of the flash block to decide which flash block is selected to be erased [16].

Equally important, since each memory cell in a flash block has a limited number of erase cycles (before the cell becomes unusable), GC also significantly affects the reliability (i.e., endurance) of SSD-based storage systems. Therefore, how to address the performance and reliability issues caused by GC of SSDs has become a critically important challenge when deploying flash-based SSDs in HPC and enterprise storage systems [19], [41].

Figure 3 shows the 3 steps of a typical GC process in SSDs, that is: (1) select the victim flash block to be erased; (2) read the valid data pages in the victim flash block and write them to other free flash blocks; (3) erase the victim flash block to mark it a free and available one for subsequent write data. Among the 3 steps, the erase latency of the flash block is the largest, usually at the *ms* level, which is much greater than the read and write latency of the flash page, usually at the *us* level.

With the development and application of the ultra-low latency Z-NAND and XL-Flash technologies that greatly amplifies the hash compute latency of the data deduplication process as it lies on the write critical path, it is no longer advisable to directly apply data deduplication to flash-based SSDs based on such technologies. To address the problem, we propose a Content-Aware Garbage Collection scheme, called CAGC, to embed the data deduplication into the data movement workflow of the GC process in ultra-low latency flash-based SSDs. The main idea behind CAGC is to hide the hash compute overhead by computing hash values of data chunks in parallel with the relatively long-latency operations of valid-page copying and flash block erase, thus alleviating the performance degradation induced by the inline data deduplication operations. To further reduce data writes, CAGC divides and writes data pages into hot or cold region based on

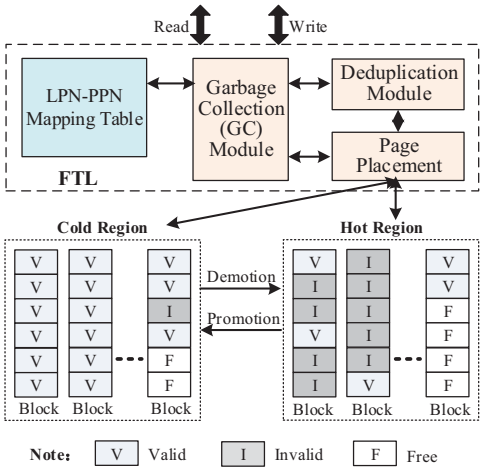


Fig. 4. Systems architecture of CAGC.

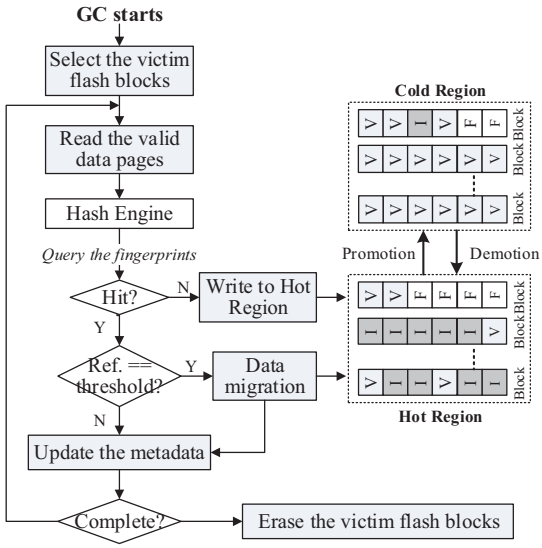


Fig. 5. The workflow of content-aware garbage collection.

their reference counts, which improves the GC efficiency and reduces the write amplification.

III. Design of CAGC

In this section, we first present a system overview of the proposed Content-Aware Garbage Collection (CAGC), followed by a description of workflow in CAGC. The reference count-based data page placement strategy of CAGC is illustrated at the end of this section.

A. System overview

The main idea behind the CAGC scheme is to remove the data deduplication operation from the foreground critical I/O path and embed it into the background GC process in ultra-low latency flash-based SSDs. CAGC not only hides the entire deduplication-induced overheads, but also avoids the negative impact of inline data deduplication on the user I/O performance. Meanwhile, CAGC leverages the characteristics of data page reference counts to separate the data pages with different reference counts into cold or hot region, which further reduces the number of valid data pages copied and the number of flash blocks erased during the GC period, thus improving both the performance and reliability of ultra-low latency flash-based SSDs.

Figure 4 shows the system architecture overview of CAGC, which mainly consists of three modules, i.e., Garbage Collection module, Data Deduplication module and Page Placement module. The Garbage Collection module is mainly responsible for selecting the victim flash blocks to be freed, migrating the valid data pages in the victim flash blocks, and then erasing these victim flash blocks. The Data Deduplication module is mainly responsible for conducting data deduplication and the valid data page migrations during the GC period to eliminate the write operations of redundant data pages, and passing the reference count information of the data pages to the Page Placement module. The Page Placement module is responsible for organizing and managing the data pages in the flash blocks according to the reference count. As shown in Figure 4, CAGC is located in the Flash Translation Layer (FTL) of SSDs. Although CAGC takes garbage collection as the one of its central component, it does not change the internal SSD GC workflow and algorithm. Therefore, CAGC is orthogonal to and easily incorporated with any existing GC algorithms, to further improve the GC efficiency.

As shown in Figure 4, the Hot Region refers to the group of flash blocks whose data pages are frequently updated to become invalid, and the Cold Region refers to the group of flash blocks whose data pages are rarely updated or deleted. When a file is deleted, the reference count of the pages related the file will only be decremented by 1 from its current reference counts. A page becomes invalid only when its reference count is reduced to 0, meaning that a data page with a high reference count will not likely be invalidated. Therefore, as shown in Figure 4, those data pages with a reference count of 1 are stored in the Hot Region, and those data pages with a higher reference count are stored in the Cold Region.

B. Workflow of content-aware garbage collection

In addition to serving user read and write requests, flash-based SSDs also need to conduct GC internally to release those flash blocks occupied by invalid data pages, so that those flash blocks can be reused for subsequent write data. Generally, flash-based SSDs utilize the system idle periods to conduct GC in the background to reclaim invalid data pages to obtain free space. However, when the free space in the SSD is lower than a preset threshold, the GC process is triggered to select the victim flash blocks that meet certain conditions specified by a given GC algorithm.

Figure 5 shows the workflow of content-aware garbage collection. When a victim flash block to be erased is selected, the valid data pages are read and the hash fingerprints of these data pages are calculated. Then CAGC searches the fingerprint in the fingerprint index to determine whether it is redundant

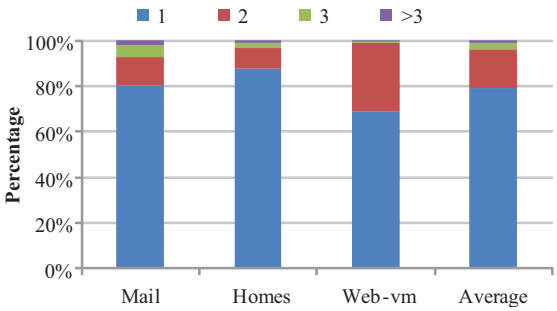


Fig. 6. The distribution of invalid pages generated from pages of different reference count.

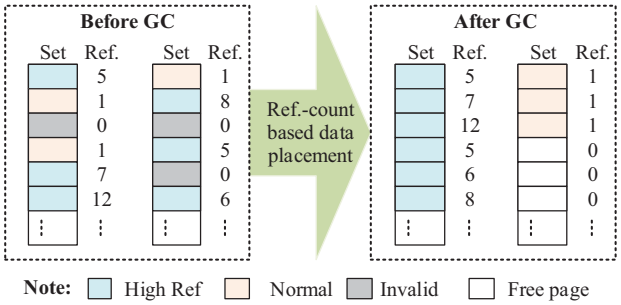


Fig. 7. An example of reference count-based data page placement.

(hit/matched) or not (missed/unmatched). If the data page is not redundant, CAGC writes the data page into the hot region and updates the fingerprint index. Otherwise, CAGC does not write the data page but simply updates the corresponding metadata, including increasing the reference count.

At the same time, when the reference count of the redundant data page is equal to the preset threshold, the data page will be migrated to the Cold Region before the victim flash block is erased. After all valid pages in the victim flash block have been migrated, the victim flash block is erased.

The flash-block erase time is generally at the millisecond level, which is much higher than the microsecond-level calculation and search overhead of the hash fingerprint. This indicates that embedding the data deduplication process into the GC process does not cause significant performance overhead. On the contrary, by using the CAGC scheme, redundant data pages will not be repeatedly written, thus improving the GC efficiency for ultra-low latency flash-based SSDs.

C. Reference count-based data page placement

The reference count of a given data page in data deduplication indicates how many different files share this data page. Intuitively, the higher the reference count for a page is, the smaller the possibility of this data page being deleted (i.e., that of all files sharing this page being deleted). It will also be verified empirically next.

Figure 6 shows the distribution of invalid data pages generated from pages of different reference count for the three FIU traces. More than 80% of invalid data pages come from flash pages with a reference count of 1, while the percentage of invalid data pages from a reference count of 3 or more is less than 1%. The analysis on these workloads shows that data pages with higher reference count have a much longer lifetime than data pages with lower reference count, and are less likely to become invalid. Therefore, data pages with high reference counts can be considered as cold data pages and stored in the Cold Region. In addition, the data pages with a reference count equal to or lower than a preset threshold (e.g., 1) can be considered as hot data pages and stored in the Hot Region.

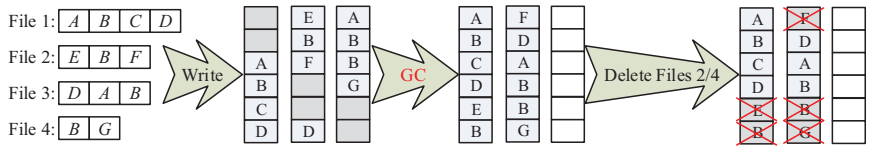
The data pages in the Hot Region have higher update frequency and higher probability of being invalid than those in the Cold Region. Therefore, the flash blocks in the Hot Region are desirable candidates for victim blocks since they are likely to contain very few valid pages that need to be migrated. Besides, each deletion or update operation on a flash data page in the Cold Region will only cause its reference count to be decremented by 1, rather than invalidating the page, meaning that the corresponding flash block will not likely have any invalid data pages. Thus, CAGC can reduce the number of valid data pages migrated and the number of flash blocks erased during GC.

Figure 7 shows an example of reference count-based data page placement, and the *Set* refers to an area composed of multiple flash blocks. In the traditional deduplication-based flash storage, data pages with different reference count are mixed and stored together. CAGC embeds the reference count-based data page placement along with the migration process of valid data pages during the GC period, thus eliminating the additional data page read and write operations from flash storage.

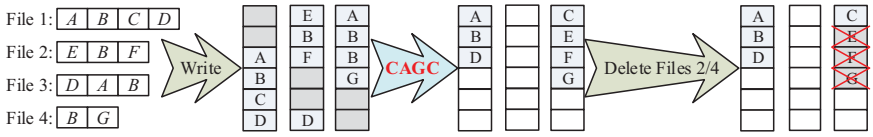
As shown in Figure 7, the reference count information of each data page can be obtained from the fingerprint index in deduplication-based storage systems. Then the reference count will be compared with a preset reference count threshold (e.g., 1). If it is larger than the threshold, the data page will be stored in the Cold Region. Otherwise, it will be stored in the Hot Region. By exploiting the reference count feature of data deduplication to guide the data page placement and leveraging the capacity optimization advantages of the data deduplication technology, the performance and reliability of the ultra-low latency flash-based SSDs can be further improved.

Figure 8 shows an example of the comparison between the traditional GC scheme and the CAGC scheme of writing 4 files and then deleting 2 of them. In the traditional SSD GC scheme, since the content redundancy of data pages is not known, data pages with the same content are redundantly stored. After some files are deleted or updated, invalid data pages appear in many different flash blocks. Therefore, it needs to migrate many more valid data pages and erase many more flash blocks than CAGC during the GC period.

As shown in Figure 8(a), the traditional GC process requires 12 valid data page write operations and 2 flash block erase operations, but only 6 flash data pages are freed. By contrast, CAGC can conduct data deduplication for the migrated data pages during the GC period, thus eliminating redundant flash



(a) An example of traditional garbage collection



(b) An example of content-aware garbage collection

Fig. 8. An example of writing four files and deleting two files between the two garbage collection schemes.

TABLE I
THE CONFIGURATION OF SSD

Type	Value	Type	Value
Page Size	4KB	Read	12us
Block Size	256KB	Write	16us
OP Space	7%	Erase Delay	1.5ms
Capacity	80GB	Hash	14us
Workloads	FIU [9]	GC Watermark	20%

page write operations. As shown in Figure 8(b), CAGC only needs 7 valid data page write operations and 1 flash block erase operation in the GC process, and 11 flash data pages can be freed. Compared with the traditional SSD garbage collection scheme, CAGC can significantly improve the GC efficiency.

IV. PERFORMANCE EVALUATIONS

In this section, we first describe the experimental setup and methodology. Then we evaluate the performance and effectiveness of our proposed CAGC scheme driven by deduplicating workloads (i.e., workload traces collected from real production systems but instrumented to enable content identification as explained later), including the comparison in the number of flash blocks erased and the number of pages written during GC. We present the testing and analysis of sensitivity study at the end of this section.

A. Experimental setup and methodology

We implement a prototype system of CAGC, which is extended on the basis of an event-driven simulator for flash-based SSDs, FlashSim [20]. It has built in various FTL management strategies and can generate response time, number of flash blocks erased and many additional performance statistics. Based on the performance parameters of Samsung's commercially available Z-NAND flash memory, the specific parameter settings of the ultra-low latency flash-based SSD in the experiment are shown in Table I.

We replay the deduplicating workloads to evaluate the performance of the CAGC prototype system. In the trace-driven experiments, the three traces were obtained from the

TABLE II
THE WORKLOAD CHARACTERISTICS

Traces	Write Ratio	Dedup. Ratio	Aver. Req. Size
Mail	69.8%	89.3%	14.8KB
Homes	80.5%	30.0%	13.1KB
Web-vm	78.5%	49.3%	40.8KB

SyLab of FIU [22] and cover a duration of three weeks. They were collected from a virtual machine running a file server (Homes), two web-servers (Web-vm) and an email server (Mail), respectively. Each request in the traces includes the hash value of the requested data. The characteristics of the three traces are shown in Table II [9], [22]. The FIU workloads with fingerprints have been widely used in the storage community to study the performance of deduplication-based storage systems.

In the experiments, we compare CAGC with the ultra-low latency flash-based SSDs without embedding the inline data deduplication during the GC process (Baseline), and the ultra-low latency flash-based SSDs with inline data deduplication embedded on the foreground user write path (Inline-Dedupe). By default, almost all the experiments in this paper are based on the greedy algorithm to select the victim flash block for all the schemes. The sensitivity study on different victim flash block selecting algorithms is presented and analyzed in Section IV-C.

B. Performance result and analysis

Figure 9 compares CAGC with the Baseline system in terms of the number of flash blocks erased, driven by the three deduplicating workloads. CAGC erases significantly smaller numbers of flash blocks than the non-deduplication Baseline system, by 23.3%, 48.3%, and 86.6%, under the Homes, Web-vm, and Mail workloads, respectively. CAGC performs data deduplication during the data page migration process of the SSD GC periods, which reduces the writing of redundant data pages and further reduces the number of erased flash blocks. For the Mail workload with highest deduplication ratio, CAGC reduces the largest number of erased flash blocks.

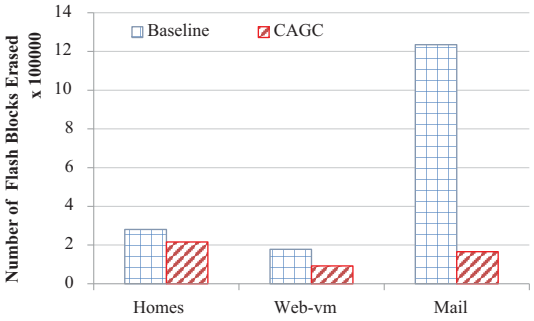


Fig. 9. A comparison of the number of flash blocks erased.

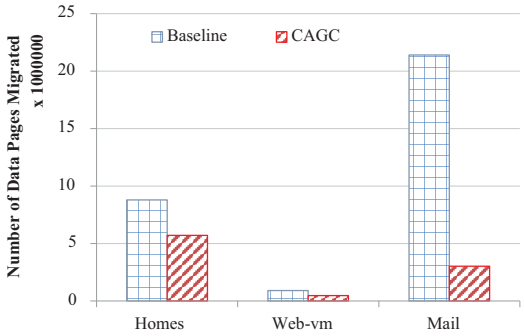


Fig. 10. The number of data pages migrated during GC.

Moreover, CAGC separates the flash data pages based on different reference counts and stores them in different regions, i.e., hot region and cold region. The greedy algorithm always selects the victim flash block that contains the most invalid data pages, thus CAGC further reduces the number of valid data page migrations during GC periods and greatly reduces the number of erased flash blocks.

CAGC reduces the number of flash blocks erased during GC because of the following two reasons. First, CAGC is based on data deduplication that significantly reduces the redundant data page write operations during GC periods. Figure 10 shows a comparison of the number of data pages migrated during GC periods driven by the three workloads. As shown in Figure 10, compared with the Baseline system, CAGC reduces the number of data pages migrated by 35.1%, 47.9%, and 85.9%, under the Homes, Web-vm, and Mail workloads, respectively. Figure 10 shows that for the Mail workload, CAGC greatly reduces the number of data pages migrated. The reason is that the data deduplication ratio of the Mail workload is over 90%, which indicates that CAGC can avoid a large number of data pages migrated during GC.

Second, the reference count-based data page placement in CAGC can effectively separate the hot and cold data pages and store them in different flash regions, thus significantly reducing the number of data pages migrated and the number of flash blocks erased during GC periods.

It is worth noting that CAGC's notable improvement on the GC efficiency, by reducing the numbers of blocks erased

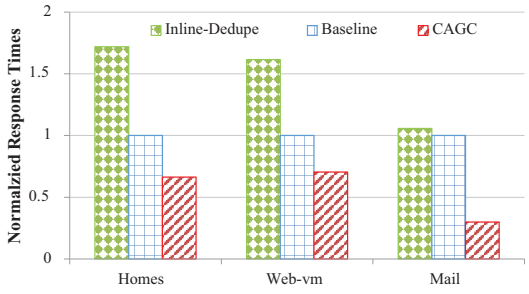


Fig. 11. A comparison of the normalized average response times.

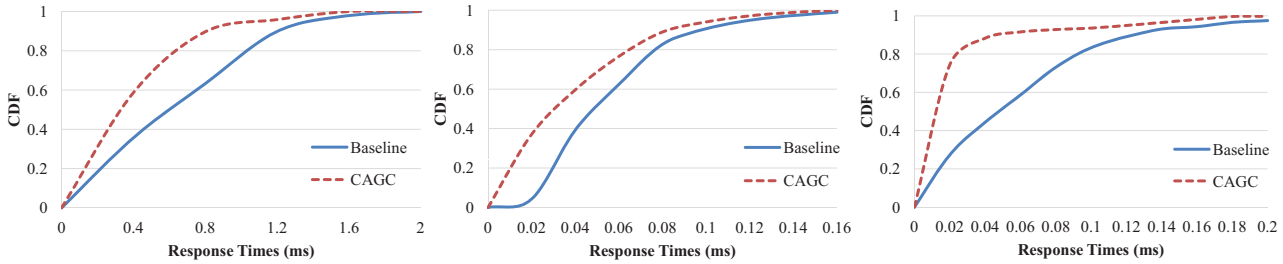
and data pages migrated, can greatly minimize GC's negative impact to application's request response time [38]. This is because GC operations in flash-based SSDs, including flash blocks erase and migration of valid pages, are very time-consuming background tasks that content for SSD internal bandwidth and directly interfere with the foreground user I/O requests.

In order to study the performance impact of different GC schemes on the average response times, Figure 11 shows a comparison of the normalized average response times during the SSD GC periods driven by the three deduplicating workloads. Compared with the Baseline system, CAGC reduces the average user response times during GC periods by 33.6%, 29.6%, and 70.1%, under the Homes, Web-vm, and Mail workload, respectively.

The main reasons are twofold. First, CAGC improves the GC efficiency by applying data deduplication during the migration of valid data pages to eliminate the write operations of redundant data pages. Thus, fewer but invalid-page-filled flash blocks are erased, meaning that the same amount of free space is claimed by a smaller number of erased blocks. Moreover, resource contention among user requests is also alleviated. User read and write requests can occupy more SSD internal resources (bandwidth). Therefore, CAGC effectively reduces the performance impact of GC operations on user read and write requests during GC periods.

Second, by improving the GC efficiency, CAGC also significantly shortens the GC duration. Since the migration of valid data pages and the erasing of flash blocks take a long time during GC periods, user read and write requests are significantly affected by the GC operations. By using data deduplication technology and reference count-based data page placement, CAGC significantly reduces the number of data pages migrated and flash blocks erased, thus speeding up the GC process. Meanwhile, CAGC also reduces the time length when user performance is degraded. In general, by both reducing the user request response time during GC periods and shortening the GC duration, CAGC significantly improves the user performance.

As shown in Figure 11, for the Mail workload, CAGC has the lowest user response time. Figure 11 also shows that inline data deduplication degrades the user performance for ultra-low latency flash-based SSDs. Especially for the Homes



(a) Homes (b) Web-vm (c) Mail

Fig. 12. Cumulative distribution function (CDF) of the response times for the Baseline system and CAGC schemes driven by the three deduplicating workloads, where the X-axis indicates the request response time and the Y-axis indicates the sum of the distribution up to a given corresponding value on the X-axis.

and Web-vm workloads with moderate data redundancy, the response times have increased by more than 50%. Ultra-low latency flash-based SSDs are generally used in primary storage systems, such as the enterprise-level and data center servers where the data deduplication ratio is moderate.

Therefore, although applying inline data deduplication on these ultra-low latency flash-based SSDs can save storage capacity and reduce the write traffic, it also significantly increases the user response time and thus reducing the storage system performance. This is counter to the purpose of using ultra-low latency flash-based SSDs in these environments, and users cannot tolerate it. Different from inline data deduplication, CAGC applies data deduplication in the GC process for ultra-low latency flash-based SSDs, which can not only reduce the impact of data deduplication on the SSD performance, but also improve the GC efficiency and storage efficiency.

In modern large-scale storage systems, such as Google, Facebook and Amazon, the long tails of the service latency have received particular attention [8]. With the wide deployment of flash-based storage devices in large-scale storage systems, the tail latency of flash-based SSDs caused by GC operations ought to be a very important consideration for the design of flash-based storage systems [12], [39].

To investigate the impact of CAGC on tail latency under different workloads, we plot the Cumulative distribution function (CDF) of the response times for the Baseline system and CAGC scheme in Figure 12 driven by the three workloads. CAGC consistently and significantly outperforms the Baseline system in terms of the tail latency performance. Especially, the response time efficiency under the mail workload is significantly higher than the other two workloads. The main reason is that under the other two workloads, the number of data pages migrated and blocks erased are relatively small and the difference is not much, leading the improvement of GC performance weak comparing the mail workloads. For example, Figure 12(c) shows that CAGC completes 80% user requests within 0.02us, while the Baseline system completes 80% user requests between 0.08us to 0.1us. Meanwhile, Since the tail latency is mainly caused by the GC operations of flash-based storage devices [39], improving the GC efficiency directly reduces the tail latency. By embedding the data deduplication process into the GC process, CAGC not only reduces flash block erase count during GC periods, but also significantly reduces the GC length, which directly alleviates the user I/O performance degradations. As a result, CAGC reduces the percentage of requests with long latency, especially for deduplicating workloads with high data deduplication ratios, such as the Mail workload.

C. Sensitivity testing and analysis

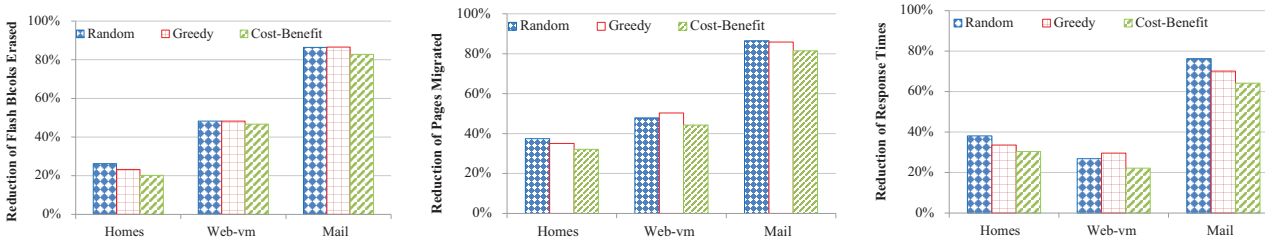
The performance of CAGC will be affected by several design parameters, such as the selection algorithm (i.e., Random, Greedy, and Cost-Benefit algorithm) for victim flash blocks. Figure 13 shows a comparison of CAGC's optimization results under different victim flash block selection algorithms in terms of number of flash blocks erase count, number of pages migrated during GC, and the average response time.

Compared with the Baseline system, under all the three flash block selection algorithms, CAGC effectively reduces the number of flash blocks erased and the number of valid data pages migrated during GC period. At the same time, CAGC effectively reduces the average user response time. The reason is that CAGC uses data deduplication to exploit data redundancy in primary storage systems, and uses the data reference count characteristics to effectively separate the hot and cold flash data pages, which greatly improves the GC efficiency of ultra-low latency flash-based SSDs. Moreover, CAGC can be easily applied to different SSD GC algorithms, such as different selection algorithms for victim flash blocks.

V. Related Work

GC is very time-consuming and thus a key performance bottleneck of flash-based SSDs. For traditional flash memory to flash-based SSDs, many studies have been conducted to optimize the GC efficiency or alleviate the GC-induced performance degradation [3], [19], [24], [31], [36], [38], [39], [41]. Generally speaking, existing studies addressing the GC-induced problems in flash-based SSDs can be classified into three categories, namely, optimizing the GC algorithms, optimizing the GC workflow and reducing the write traffic to flash.

First, optimizing the GC algorithms can directly improve the GC efficiency of flash memory. These optimizations usually focus on different steps of GC algorithms [10], [40]: when to trigger the GC process, which flash block is selected to be



(a) Number of flash blocks erased (b) Number of date pages migrated during GC (c) Response time

Fig. 13. A comparison of the optimization results under different victim flash block selection algorithms in terms of the number of flash blocks erased, the number of data pages migrated during GC, and the average response times.

erased, and how to merge the valid data pages, etc. Among these proposed schemes, how to identify and store the hot and cold data pages in different flash blocks is the key to improve the GC efficiency. Different from the previous studies that exploit the spatial locality based on logic block addresses, CAGC exploits the content locality, expressed by the reference counts in deduplicaiton-based memory systems, to identify the hot and cold data pages. Thus, CAGC effectively reduces the number of valid data pages migrated, which directly improves the GC efficiency [24].

Second, due to the long tail latencies of GC operations, user read and write requests are significantly affected by the on-going flash block erase operations. To alleviate the GC-induced user performance degradation, schemes such as semi-preemption GC [23], erase suspension [35] and partial-erase [24] are proposed to give higher priority to serve user I/O requests. By suspending the on-going flash block erase operations, internal flash resources can be allocated to serve user I/O requests much more efficiently. However, all these schemes only change the GC workflow to alleviate the user performance degradation but do not improve the GC efficiency.

Third, GC operations are also affected by the write data volume. Reducing the write data can directly reduce GC operations. Thus, schemes such as write buffer with hard drive or non-volatile memory [32], [36], inline data compression [25] and inline data deduplication [2], [11], [26] are applied to reduce the write traffic of flash-based SSDs. However, with the emergence of ultra-low latency flash-based SSDs, inline data reduction with its operations lying on the critical I/O path will introduce significant performance overhead for ultra-low latency flash-based SSDs. By contrast, CAGC embeds the data deduplication into the GC process to hide the deduplication-induced performance overhead by exploiting parallelism, while reaping the benefits of deduplication-introduced data reduction.

VI. Conclusion

With the advent of Samsung's Z-NAND and Toshiba's XL-Flash technologies, directly applying inline data deduplication in these ultra-low latency flash-based SSDs can degrade the storage performance. To address the problem, this paper proposes a Content-Aware Garbage Collection technology (CAGC) that embeds the data deduplication into the GC process to hide the performance overhead. Meanwhile, CAGC uses the reference count-based data page placement to exploit the reference count feature in deduplicating storage systems, which effectively separates the hot and cold data pages. Thus, CAGC can further reduce the number of data page write operations and the block erase count during the SSD garbage collection period. The performance results on a CAGC prototype implemented in FlashSim show that CAGC effectively reduces the numbers of flash blocks erased and valid pages migrated during the SSD GC period, thus further improving the user I/O performance and reliability for ultra-low latency flash-based SSDs.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. U1705261, No. 61872305, and No. 61972325, the US NSF under Grant No. CCF-1704504 and CCF-1629625.

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, and R. Panigrahy. Design Tradeoffs for SSD Performance. In *Proceedings of the 2008 USENIX Annual Technical Conference (USENIX'08)*, pages 57–70, Boston, MA, Jun. 2008.

[2] F. Chen, T. Luo, and X. Zhang. CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11)*, pages 77–90, San Jose, CA, Feb. 2011.

[3] Z. Chen and Y. Zhao. DA-GC: A Dynamic Adjustment Garbage Collection Method Considering Wear-Leveling for SSD. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI (GLSVLSI'20)*, pages 475–480, Sep. 2020.

[4] W. Cheong, C. Yoon, S. Woo, K. Han, D. Kim, C. Lee, Y. Choi, S. Kim, D. Kang, G. Yu, J. Kim, J. Park, K. Song, K. Park, S. Cho, H. Oh, D. Lee, J. Choi, and J. Jeong. A Flash Memory Controller for 15us Ultra-Low-Latency SSD Using High-Speed 3D NAND Flash with 3us Read Time. In *Proceedings of the 2018 International Solid-State Circuits Conference (ISSCC'18)*, pages 338–340, San Francisco, CA, Feb. 2018.

[5] J. Colgrove, J. Davis, J. Hayes, E. Miller, C. Sandvig, R. Sears, A. Tamches, N. Vachharajani, and F. Wang. Purity: Building Fast, Highly-Available Enterprise Flash Storage from Commodity Components. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD'15)*, Melbourne, Victoria, Jun. 2015.

[6] Data Reduction in Pure Storage. https://www.purestorage.com/products/purity/purity-reduce.html. 2017.

[7] Datacenters Flush with All-Flash Storage. https://www.enterpriseai.news/2019/11/11/datacenters-flush-with-all-flash-storage/. Nov. 2019.

[8] J. Dean and L. Barroso. The Tail at Scale. *Communications of the ACM*, 56(2):74–80, 2013.

[9] FIU IODedup traces. http://iotta.snia.org/traces/391.

[10] E. Gal and S. Toledo. Algorithms and data structures for flash memories. *ACM Computing Surveys*, 37(2):138–163, 2005.

[11] A. Gupta, R. Pisolkar, B. Urgaonkar, and A. Sivasubramaniam. Leveraging Value Locality in Optimizing NAND Flash-based SSDs. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11)*, pages 91–103, San Jose, CA, Feb. 2011.

[12] M. Hao, G. Soundararajan, D. Kenchammana-Hosekote, A. A. Chien, and H. S. Gunawi. The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST'16)*, Santa Clara, CA, Feb. 2016.

[13] B. Harris and N. Altiparmak. Ultra-Low Latency SSDs' Impact on Overall Energy Efficiency. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'20)*, Jul. 2020.

[14] M. Jung, W. Choi, J. Shalf, and M. Kandemir. Triple-A: A Non-SSD Based Autonomic All-Flash Array for Scalable High Performance Computing Storage Systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'14)*, Salt Lake City, UT, Apr. 2014.

[15] M. Jung, R. Prabhakar, and M. Kandemir. Taking Garbage Collection Overheads Off the Critical Path in SSDs. In *Proceedings of the ACM/IFIP/USENIX 13th International Middleware Conference (Middleware'12)*, Montreal, Canada, Dec. 2012.

[16] A. Kawaguchi, S. Nishioka, and H. Motoda. A Flash-Memory Based File System. In *USENIX 1995 Technical Conference*, New Orleans, LA, jan. 1995.

[17] A. Khanbadr, M. B. Marvasti, S. A. Asghari, S. Khanbadr, and A. M. Rahmani. A Novel Method for Victim Block Selection for NAND Flash-based Solid State Drives Based on Scoring. *The Journal of Supercomputing*, 76(9), 2020.

[18] J. Kim, C. Lee, S. Lee, I. Son, J. Choi, S. Yoon, H. Lee, S. Kang, Y. Won, and J. Cha. Deduplication in SSDs: Model and Quantitative Analysis. In *Proceedings of the 28th International Conference on Massive Storage Systems and Technologies (MSST'12)*, pages 91–103, Apr. 2012.

[19] J. Kim, K. Lim, Y. Jung, S. Lee, C. Min, and S. H. Noh. Alleviating Garbage Collection Interference Through Spatial Separation in All Flash Arrays. In *Proceedings of the 2019 USENIX Annual Technical Conference (ATC'19)*, pages 799–812, Renton, WA, Jul. 2019.

[20] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar. FlashSim: A Simulator for NAND Flash-Based Solid-State Drives. In *Proceedings of the 2009 First International Conference on Advances in System Simulation*, pages 125–131, Porto, Portugal, Oct. 2009.

[21] S. Koh, C. Lee, M. Kwon, and M. Jung. Exploring System Challenges of Ultra-Low Latency Solid State Drives. In *Proceedings of the 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'18)*, Boston, MA, Jul. 2018.

[22] R. Koller and R. Rangaswami. I/O deduplication: Utilizing content similarity to improve I/O performance. *ACM Transactions on Storage (TOS)*, 6(3):1–26, 2010.

[23] J. Lee, Y. Kim, G. Shipman, S. Oral, F. Wang, and J. Kim. A Semi-Preemptive Garbage Collector for Solid State Drives. In *Proceedings of the 2011 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'11)*, pages 12–21, Austin, TX, Apr. 2011.

[24] C.-Y. Liu, J. B. Kotra, M. Jung, and M. T. Kandemir. PEN: Design and Evaluation of Partial-Erase for 3D NAND-Based High Density SSDs. In *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST'18)*, Oakland, CA, Feb. 2018.

[25] B. Mao, H. Jiang, S. Wu, Y. Yang, and Z. Xi. Elastic Data Compression with Improved Performance and Space Efficiency for Flash-based Storage Systems. In *Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS'17)*, pages 1109–1118, Orlando, FL, May 2017.

[26] B. Mao, S. Wu, H. Jiang, X. Chen, and W. J. Yang. Content-aware Trace Collection and I/O Deduplication for Smartphones. In *Proceedings of the 33rd International Conference on Massive Storage Systems and Technology (MSST'17)*, Santa Clara, CA, May 2017.

[27] J. Meza, Q. Wu, S. Kumar, and O. Mutlu. A Large-Scale Study of Flash Memory Failures in the Field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'15)*, pages 177–190, Portland, Oregon, USA, Jun. 2015.

[28] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In *Proceedings of the 4th European Conference on Computer Systems (EuroSys'09)*, pages 145–158, Nuremberg, Germany, Mar. 2009.

[29] S. W. Lee and D. J. Park and T. Sun. Chung and D. H. Lee and S. Park and H. J. song. A log buffer-based flash translation layer using fully-associative sector translation. *Acm Transactions on Embedded Computing Systems*, 2007.

[30] B. Schroeder, R. Lagisetty, and A. Merchant. Flash Reliability in Production: The Expected and the Unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST'16)*, pages 67–80, San Jose, CA, Feb. 2016.

[31] N. Shahidi, M. Arjomand, M. Jung, M. Kandemir, C. R. Das, and A. Sivasubramaniam. Exploring the Potentials of Parallel Garbage Collection in SSDs for Enterprise Storage Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16)*, Salt Lake City, UT, Nov. 2016.

[32] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber. Extending SSD Lifetimes with Disk-Based Write Caches. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST'10)*, pages 101–114, San Jose, CA, Feb. 2010.

[33] C. W. Tsao, Y. H. Chang, and M. C. Yang. Performance Enhancement of Garbage Collection for Flash Storage Devices: An Efficient Victim Block Selection Design. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, Austin, TX, USA, jun. 2013.

[34] With Nimble, Less is More. https://www.nimblestorage.com/its-all-about-data-reduction/. 2017.

[35] G. Wu and B. He. Reducing SSD Read Latency via NAND Flash Program and Erase Suspension. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*, San Jose, CA, Feb. 2012.

[36] S. Wu, Y. Lin, B. Mao, and H. Jiang. GCaR: Garbage Collection aware Cache Management with Improved Performance for Flash-based SSDs. In *Proceedings of the 30th International Conference on Supercomputing (ICS'16)*, Istanbul, Turkey, Jun. 2016.

[37] S. Wu, J. Zhou, W. Zhu, H. Jiang, Z. Huang, Z. Shen, and B. Mao. EaD: a Collision-free and High Performance ECC assisted Deduplication Scheme for Flash Storage. In *Proceedings of the 38th IEEE International Conference on Computer Design (ICCD'20)*, Oct. 2020.

[38] S. Wu, W. Zhu, G. Liu, H. Jiang, and B. Mao. GC-aware Request Steering with Improved Performance and Reliability for SSD-based RAIDs. In *Proceedings of the 32nd IEEE International Parallel & Distributed Processing Symposium (IPDPS'18)*, Vancouver, British Columbia, Canada, May 2018.

[39] S. Yan, H. Li, M. Hao, H. Tong, S. Sundararaman, A. Chien, and H. Gunawi. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST'17)*, Santa Clara, CA, Feb. 2017.

[40] M.-C. Yang, Y.-M. Chang, C.-W. Tsao, P.-C. Huang, Y.-H. Chang, and T.-W. Kuo. Garbage Collection and Wear Leveling for Flash Memory: Past and Future. In *Proceedings of the 2014 International Conference on Smart Computing*, Hong Kong, China, Nov. 2014.

[41] P. Yang, N. Xue, Y. Zhang, Y. Zhou, L. Sun, W. Chen, Z. Chen, W. Xia, J. Li, and K. Kwon. Reducing Garbage Collection Overhead in SSD Based on Workload Prediction. In *Proceedings of the 11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'19)*, Renton, WA, Jul. 2019.

[42] J. Zhang, M. Kwon, D. Gouk, C. Lee, M. Alian, M. Chun, M. Kandemir, J. Kim N. Kim, and M. Jung. FlashShare: Punching Through Server Storage Stack from Kernel to Firmware for Ultra-Low Latency SSDs. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI'18)*, pages 477–492, Oakland, CA, Oct. 2018.