X-mean GC: A Novel Wear-leveling Scheme for NAND Flash

Tuo Ding, Yuanze Liu*

***Abstract*—NAND flash-based storage systems are expected to become the next generation of primary storage devices due to several key advantages inherent to NAND flash memory. However, the limited endurance of NAND flash remains a major obstacle to widespread adoption. To mitigate this issue, wear leveling has been extensively researched, particularly for its effectiveness in managing block erasure counts.**

This paper presents a novel garbage collection approach, the X-Mean Garbage Collection (X-Mean GC) scheme, which is simple to implement and requires minimal system resources. The X-Mean GC scheme enhances endurance by dynamically adjusting a threshold to effectively limit the erasure count of each block. Moreover, it can be seamlessly integrated with other garbage collection algorithms. When combined with four existing algorithms, the X-Mean GC scheme achieves an impressive lifetime improvement of up to 133.35% compared to the original methods.

***Index Terms*—Garbage collection, NAND flash memory, wear-leveling.**

I. INTRODUCTION

With advancements in cloud storage and big data technologies [1], [2], the digital universe is expanding rapidly, driving an urgent demand for high-speed, high-capacity storage devices. However, traditional hard disk drives (HDDs) face inherent mechanical limitations that restrict improvements in access speed. Meanwhile, dynamic random-access memory (DRAM) stores data by charging and discharging capacitors, but as DRAM cells shrink, their ability to retain electric charge diminishes, leading to reliability challenges that hinder large-scale integration [3], [4].

In contrast, NAND flash-based storage systems offer several advantages, including low power consumption, minimal noise, faster access speeds, and improved seismic performance [5]–[8]. Unlike HDDs, which rely on magnetic media, NAND flash utilizes floating-gate field-effect transistors. Data programming occurs when electrons are injected into the floating gate of each memory cell via Fowler–Nordheim tunneling, while erasure is achieved by releasing these electrons, thereby enhancing reliability. However, each program/erase (P/E) cycle degrades the tunnel oxide layer in the transistors, significantly limiting NAND flash's lifespan compared to magnetic storage.

Manuscript received January 8, 2025; revised July 4, 2025.

Tuo Ding is a manager of the National Minorities Energy and Technology Co., Ltd., Beijing 100000 China (corresponding author to provide phone: 17710780831; e-mail: tour1988@126.com).

Yuanze Liu is a PhD candidate in the College of Electronic Information and Optical Engineering, Nankai University, Tianjin 300000 China (e-mail: 16600297947@163.com).

This endurance challenge [9] becomes even more pronounced in high-capacity NAND flash systems, where increasing cell densities and thinner tunnel oxide layers accelerate wear. For example, early NAND flash storage systems using single-level cells (SLCs) could endure approximately 100,000 P/E cycles, whereas modern triple-level cell (TLC) NAND flash is typically limited to around 2,500 P/E cycles [10], [11].

Wear-leveling techniques are designed to address endurance challenges and ensure that NAND flash-based storage systems maintain stable performance throughout their lifespan. A key metric for evaluating wear-leveling effectiveness is the erasure count of frequently accessed (*hot*) blocks. As a block's erasure count approaches the upper limit of program/erase (P/E) cycles, errors may accumulate within the stored data, compromising reliability. The fundamental principle of wear-leveling is to distribute write and update operations evenly across the NAND flash memory, thereby preventing premature wear of individual memory cells [12]–[16].

Recent studies on wear-leveling [17]–[19] commonly classify data as *hot* or *cold* based on access frequency. Memory cells storing hot data experience accelerated wear compared to those holding cold data, leading to uneven erasure counts across blocks. To address this imbalance, hot data is frequently migrated to blocks with lower erasure counts. However, such approaches introduce additional read, write, and erase operations, which can degrade both system performance and endurance. These limitations primarily arise from garbage collection algorithms that lack adaptive intelligence. Given the pivotal role of garbage collection in wear-leveling, designing an optimized garbage collection algorithm is essential to achieving a balance between endurance and performance.

The out-of-place update mechanism in NAND flash-based memory [20] generates numerous invalid pages during operation. Garbage collection reclaims storage space by erasing blocks containing these invalid pages [21]. To minimize overhead, traditional approaches select blocks with the fewest valid pages as *victim blocks* for garbage collection [22]. This method reduces garbage collection response time but does not effectively support wear-leveling.

Several algorithms attempt to balance garbage collection efficiency with wear-leveling. The cost-benefit (CB) algorithm and cost-age-time (CAT) algorithm [23] incorporate block age information to differentiate between hot and cold data. While these methods improve wear-leveling, they require additional hardware timers, imposing significant overhead on NAND flash-based storage systems. Other approaches, such

as the write-order-based garbage collection (WO_GC) [24] and the wear-conscious garbage collection scheme (WECO) [25], integrate erasure counts by calculating scores based on a block's individual erasure count (N_{erase}) relative to the maximum erasure count among all blocks ($Max.N_{erase}$). These scores prioritize garbage collection, but the non-linear relationship between score and N_{erase} results in a small number of hot blocks accumulating disproportionately high erasure counts, ultimately undermining wear-leveling efforts.

Although existing research has mitigated wear-leveling issues to some extent, no effective method explicitly limits erasure counts. Algorithms that consider erasure counts [26], [27] suffer from non-linear score-to-erasure count relationships, allowing some blocks to experience excessive wear. This lack of precise control disrupts wear-leveling and reduces NAND flash system stability. Since erasure counts continuously evolve, fixed thresholds for restricting erasure counts are impractical. Instead, an adaptive reference point for dynamically managing erasure counts is both necessary and crucial for enhancing NAND flash endurance.

In this paper, we propose a novel wear-leveling scheme, the X-mean garbage collection (X-mean GC) scheme. This scheme involves building a candidate pool, where candidate blocks are selected based on the average erasure count of all blocks in NAND flash-based memory. The average erasure count is updated in real-time, ensuring that the candidate pool consistently maintains a sufficient number of candidates. To achieve effective wear-leveling, we limit the erasure counts of candidate blocks within the pool to a dynamic threshold that adapts to current conditions. The main contributions of this manuscript are as follows:

- We introduce a novel wear-leveling scheme, X-mean GC, based on a dynamically maintained candidate pool.
- We integrate X-mean GC with four existing garbage collection algorithms, each showing improved wear-leveling performance after modification.
- We analyze the impact of the parameter X on both wear-leveling performance and system overhead, and identify an optimized X value based on a comprehensive evaluation.

The remainder of this paper is organized as follows: Section II provides a brief review of related work on existing wear-leveling schemes. Section III describes the architecture of the proposed X-mean GC scheme. Section IV is divided into four parts: Section IV-A outlines the experimental simulation environment; Section IV-B introduces the lifetime model used to evaluate the lifespan of NAND flash-based storage systems under various wear-leveling strategies; Sections IV-C and IV-D present the experimental results focused on different objectives. Finally, we conclude our work in Section V.

II. RELATED WORK

The purpose of X-mean GC is to build a new mechanism to strictly restrict the erasure count of each block among NAND flash-based memory to a reasonable bound. In this section, we briefly describe the existing garbage collection algorithms and wear-leveling schemes.

As the object of garbage collection is block [28], a good garbage collection algorithm should take different characteristics of blocks into full consideration. Firstly, the ratio of valid page in a block influences the efficiency of the garbage collection. The garbage collection course of a block with more valid pages involves more read and write operations and certainly consumes more time. As a result, less free space is released in this course. Secondly, the P/E cycles of every block are finite. A high erasure count of block means this block is nearly worn out and impacts the performance of NAND flash-based storage system. Thirdly, old age blocks which haven't been updated for a long time will most probably not be updated in the near future because most data have time locality. Data in old age blocks can also be called cold data and they always contain more errors compared with hot data. In the meantime, avoiding cold blocks being released for a long time will enlarge the difference of the erasure count of hot blocks, which is harmful to wear-leveling. These guidelines can also be found in the researches we introduce below.

In early garbage collection algorithms, only the ratio of valid page is considered. Wu et al. [29] proposed a greedy garbage collection scheme in 1994. In greedy garbage collection scheme, the garbage collection algorithm only select the block with least valid pages to be the victim. Greedy is a simple and efficient solution, but it is far away from a good wear-leveling performance. The cost benefit (CB) garbage collection scheme [30] combines age information and ratio of valid page together, and the score of a block is calculated according to the two factors. Garbage collection course will erase the block with the highest score when the number of free blocks is below the assigned threshold. In order to calculate the age, a timer to record the system time is needed. The age of a block is calculated by the current time minus the previous update time. As the lifetime of a NAND flash-based storage system can reaches several years, the ages of blocks may be a series of huge numbers which occupy a lot of memory space and DRAM space for storing them. Despite the defects above, the CB scheme shows good performance in efficiency and lifetime.

The cost-age-times (CAT) garbage collection algorithm [31] adds the erasure count into its expression of the score which is shown in Table I. Owe to the effect of the erasure count, the CAT algorithm performs better in terms of lifetime than the CB scheme.

The fast and efficient garbage collection (FeGC) algorithm [23] takes the sum of all invalid pages' age in a block as the score to choose the victim block, which is an efficient and novel garbage collection scheme. The way FeGC uses is helpful to raise the accuracy of distinguishing old blocks and is certainly good to prolong NAND flash-based storage system's lifetime. However, the granularity of age information used in FeGC is smaller and thus the age information occupies more memory space than CB and CAT.

Swap-aware garbage collection policy (SG) [32] is a garbage collection policy used in flash-memory-based swap

system. Flash-memory-based swap system treats NAND flash memory as swap area and this policy can alleviate the huge performance gap between DRAM and HDD. However, this brings in a lot of I/O operations which adds a great burden to the NAND flash memory. SG takes full account of the characteristics of NAND flash and shows a good performance in terms of wear-leveling.

Swap-aware garbage collection policy (SCATA) [33] and swap time-aware garbage collection policy (STGC) [34] both take the swap time of the swap system into account. Due to this new characteristic, they can reduce the time consumption of garbage collection effectively with good wear-leveling performance simultaneously.

As the key role of the erasure count information playing in wear-leveling, the wear-conscious garbage collection (WECO) scheme [25] inclines more towards it. As shown in Table I, there are more parameters related to the erasure count in the expression of the score. WECO takes not only the erasure count of a single block into account, but also takes the Maximum and Minimum erasure count among all blocks in NAND flash-based memory into consideration. Thus, WECO could control the intensity of wear-leveling depending on current wear conditions. Comparing with the greedy algorithm,

this scheme makes a good trade-off between the performance and the wear-leveling. The write order-based garbage collection (WO-GC) scheme [24] is an advanced garbage collection algorithm which take all guidelines into account. Different from other algorithms considering the age information, WO-GC doesn't need an extra timer. In the WO-GC scheme, age information is tracked by WSN (write sequence number), which means the order to receive the latest write request for the block. Owe to WSN, WO-GC performs better in power consumption and memory space occupation.

Adaptive wear-leveling [35] keeps a table to record erasure counts of all data blocks, and this list is used to identify the blocks that have hot or cold data as well. Adaptive wear-leveling detects the current status of the difference between the maximum erasure count and the minimum erasure count among all data blocks. If the difference exceeds the threshold, adaptive wear-leveling will be triggered. Adaptive wear-leveling depends on cold data migrations to achieve prolonging lifetime, so it will lead to bigger write amplification.

DWARAM [36] divides the non-volatile memory (NVM) pages into different wear ranges according to their write count. Every wear range has its own maximum write count and minimum write count, and all pages' write counts in this range fall between these two extremes. The divided ranges can reduce the retrieve overhead and performs better response time of garbage collection. There is a limit to the number of pages in a range in order to keep the retrieve course efficient enough. If there is a range exceeds the limit number of pages, this range will be split into two new ranges according to the write counts' median of the original range. If there is a page's write count bigger than all range's maximum write count, a new range will be created and its maximum write count equals this page's write count. DWARAM can achieve good wear-leveling performance, but it doesn't take the ratio of valid page and the age information into account.

Similar to DWARM, runtime system approach [37] also divides memory space into different ranges–generational heaps. Generational garbage collectors divide all blocks into young generation heap, hybrid generational heap and old generation heap. In order to reduce write amplification and relieve write traffic, young generation heap is stored in DRAM temporarily. Once the garbage collection course is triggered, the controller of the device will find the region with the maximum write count among the generational heaps, then valid data will be migrated to regions with smaller write count. It is worth noting that the wear-leveling course and the garbage collection course are executed at the same time. This will not bring in extra page writes.

Masaru Nakanishi et al. [38] design an application-oriented Wear-leveling algorithm for NAND flash-based storage system. Their algorithm needs to record overwrite count of each sector (WE_{sector}) and hold overwrite count of each page (WE_{th}). When NAND flash-based storage system receives a write request, the written page will compare its maximum WE_{sector} and (WE_{th}). If the latter one is bigger than the former one, the write request will be executed in this page. Otherwise, WE_{th} of this page will be added by five and the write request will be assigned to another page. In this algorithm, fine-gained overwrite count information occupies a lot of storage space and the wear-leveling performance in some extreme cases is debatable.

File-aware garbage collection algorithm (FaGC+) [39] is a novel garbage collection algorithm that divides data into four types. It can distinguish hot data and cold data according to the parameter "h" meticulously. This algorithm shows good wear-leveling degree and low overhead under the Zipf distribution.

Besides the researches mentioned above, data compression technique [40] and update data in DRAM [41]–[43] can also prolong lifetime of NAND flash-based storage system. Their main purpose is to reduce the amount of data access in NAND flash-based storage system, but the unbalance of erasure counts is still a problem they have to face. In addition, some previous works such as [44] focus on the large-scale cluster memory optimization.

III. X-MEAN GARBAGE COLLECTION SCHEME

A. OVERVIEW

As shown in Figure 1, there are two steps in X-mean GC. Firstly, all used blocks will be filtrated by X-mean GC and then be joined in the candidate pool. Secondly, victim block is selected by X-mean GC according to the age or the ratio of valid page factor from the candidate pool. We can see step one and step two are two independent steps, they have no effect on each other. The algorithm chosen by the step two is very flexible. We can choose either the existing algorithm or create a new algorithm.

TABLE I: Summary of garbage collection schemes. u: percentage of valid page, N_erase: erasure count, age: time since last data update, S_k: page sequence number of the k^{th} page

GC algorithm	Block recycling policy	Block allocation policy	Characteristic
Greedy	Choose the block with the lowest score $score = u$	FIFO	Short latency
Cost-benefit (CB)	Choose the block with the highest score $score = \frac{age(1-u)}{u}$	FIFO	Short latency Age considered
Cost-age-times (CAT)	Choose the block with the lowest score $score = \frac{u}{1-u} \times \frac{1}{f(age)} \times N_{erase}$	Youngest block first	Short latency Age considered Wear leveling considered
Swap-aware garbage collection (SG)	Choose the block with the lowest score $score = \frac{1}{u+n\times pages} \times \frac{1}{N_{erase}} \times f(age)$	Not mentioned	Short latency Age considered Erase count considered Wear leveling considered
FeGC	Choose the block with the highest score $\sum_{i=1}^{n} age_i$	Adaptive youngest block	Short latency Age considered
Write order based garbage collection (WO_GC)	Choose the block with the lowest score $score = \frac{u}{1-u} \times \frac{1}{\frac{MaxWSN-WSN}{MaxWSN}} \times \frac{N_{erase}}{MaxN_{erase}}$	FIFO	Short latency Age considered Erase count considered
FaGC+	Choose the block with the highest score $score = \sum i = 0n(D_i)$ $D_I = S_{k+1} - S_k$	According to the value of h	Age considered Erase count considered Wear leveling considered

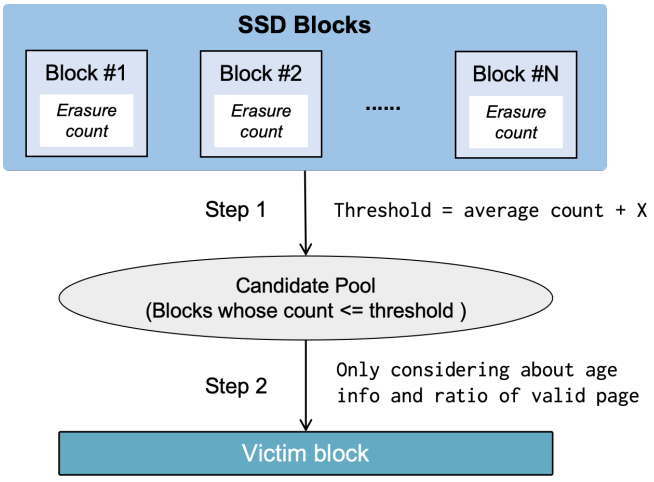


FIGURE 1. Victim block screening process of X-mean GC.

The overhead of step one is only a few registers of NAND flash-based storage system which are used to track the average erasure count of all blocks and the number of erase operation. The average erasure count and the number of erase operation of a new NAND flash-based storage system are both zero. As shown in Algorithm 1, the average erasure count is added by one whenever the number of the erase operation reaches the sum of blocks, then the number of erase operation is cleared. The overhead of step two is based on the algorithm chosen and will not be expanded here.

Algorithm 1 Calculation of Threshold

Require: number of erase operation, the average erasure count, X.
Ensure: threshold
0: **if** an erase operation comes **then**
0: number of erase operation ← number of erase operation + 1;
0: **while** number of erase operation = sum of blocks **do**
0: average erasure count ← average erasure count + 1;
0: number of erase operation ← 0;
0: **end while**
0: threshold ← average erasure count + X;
0: **end if**=0

B. THE CANDIDATE POOL

As discussed in Section I, the reference object of the threshold of the erasure count should be variable. The most ideal wear-leveling is that all blocks have the same erasure count that equals to the average erasure count of all blocks, so we choose the average erasure count as the reference object. However, the erasure count of a single block is always floating in practical use. If we can limit the fluctuation in an appropriate range which is close to the average erasure count,

TABLE II: Flash memory information

Flash memory information	
Block size	256KB
Page per block	64
Page size	4KB
Read latency	25us/page
Write latency	200us/page
Erase latency	2ms/block

TABLE III: Flash memory information

	Write request ratio	Total request count	Average request size (KB)
Financial	76.80%	5334987	4.38
Pro	89.20%	5585886	10.03
Systor	25.30%	2761269	24.42

a good wear-leveling performance can be expected. According to the analysis above, we decide to add the average erasure count by a constant X as the threshold of the erasure count. There are two extreme cases of X:

- X is too big that it can't not restrict the erasure count efficiently;
- X is too small that every erase operation will bring in a lot of page copies because there are little blocks suitable for garbage collection and they always have a high ratio of valid page.

In this paper, we set X to 10, as will be explained in Section IV-D. We call the blocks, whose erasure count is equal to or smaller than the threshold, candidate blocks. Candidate blocks compose candidate pool and wait for the next step of X-mean GC.

IV. EVALUATION

In this section, we evaluate the effectiveness of X-mean GC and answer the following questions:

- How does X-mean GC behave against other garbage collection schemes?
- How much does X-mean GC prolong the device lifetime by?
- How does the value of X affect the performance of X-mean GC?

We will firstly describe the simulation environment and the lifetime model, and then evaluate X-mean GC with three benchmarks.

A. SIMULATION ENVIRONMENT

Since our work concentrates on wear-leveling, an appropriate simulation software is needed. We select a widely used NAND flash simulation software called flashsim [45] that works under linux 10.10 as the simulation environment. Flashsim is consisted by various modules which can realize most functions of NAND flash-based storage system. Through setting key parameters, flashsim can simulate different NAND flash chips, address mapping algorithms and garbage collection algorithms. In this paper, we select the Micron chip MT29F16G08ADACA as simulation object with the detail parameters shown in Table II. Because the scheduling algorithm of channels may affect the performance of the garbage collection algorithm, our NAND flash-based storage system has only one channel, 8192 targets and a 32MB DRAM. There are three built-in address mapping algorithms (pagemap, FAST and DFTL) in flashsim. Demand-based flash translation layer (DFTL) [46] is selected to be the address mapping algorithm because it is more efficient than fully-associative sector translation (FAST) and closer to the real conditions of NAND flash-based storage system than pagemap. In order to increase the credibility, three realistic traces (financial, prn and systor) are applied in our experiment.

TABLE IV: Algorithms of step two

GC algorithm	Step two algorithm
GX-mean GC	$score = u$
CX-mean GC	$score = \frac{age(1-u)}{u}$
CAX-mean GC	$score = \frac{u}{1-u} \times \frac{1}{f(age)}$
WX-mean GC	$score = \frac{u}{1-u} \times \frac{1}{\frac{MaxWSN-WSN}{MaxWSN}}$

B. THE LIFETIME MODEL

We need a methodology to evaluate the effectiveness of X-mean GC, so we choose the lifetime model which is first used in the previous work of wear-leveling [9]. The lifetime model borrowed the idea of Gini coefficient, which is used to present the wealth distribution of a nation's residents in economics. In this model, a device is considered unusable when t% pages reach their endurance limits. The expression of the lifetime improvement (X) is as follows:

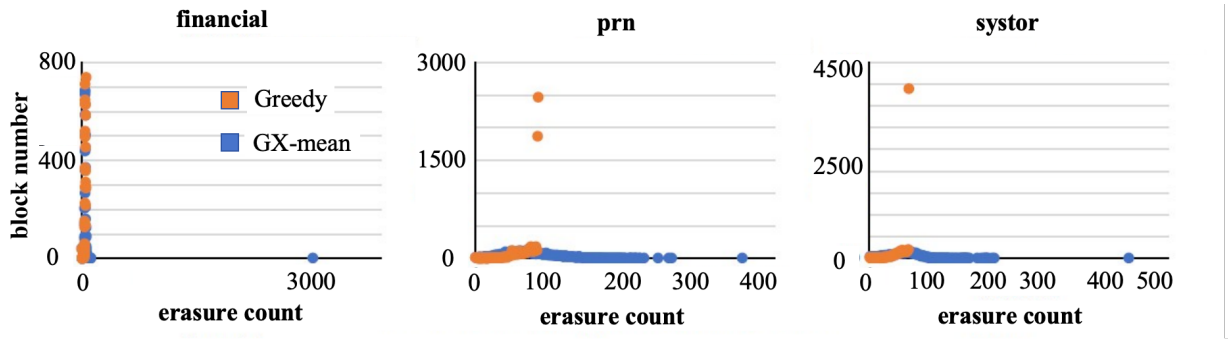
$$X = \frac{\sum_{i \in S_t} w_i}{\sum_{i \in S_t^*} w_i^*}$$

In this paper, we change pages to blocks because the unit of erase operation in NAND flash memory is block. In other words, S_t and S_t^* denote the sets of top $t\%$ hot blocks with X-mean GC and with control group algorithm, i block endures w_i and w_t^* erase operations respectively.

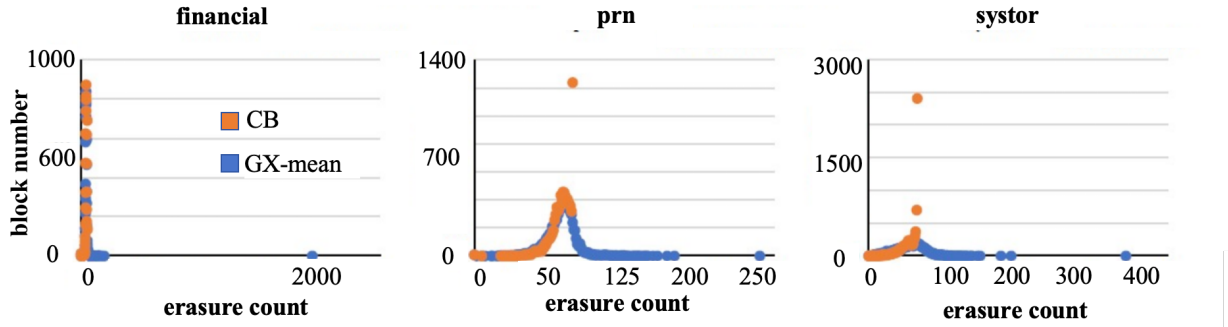
C. SIMULATION

As we illustrated in Section II, there are three key factors influencing the performance of a garbage collection scheme. In this paper, we select Greedy, CB, CAT and WO_GC as competitors of X-mean GC. They take different number of these factors into account respectively.

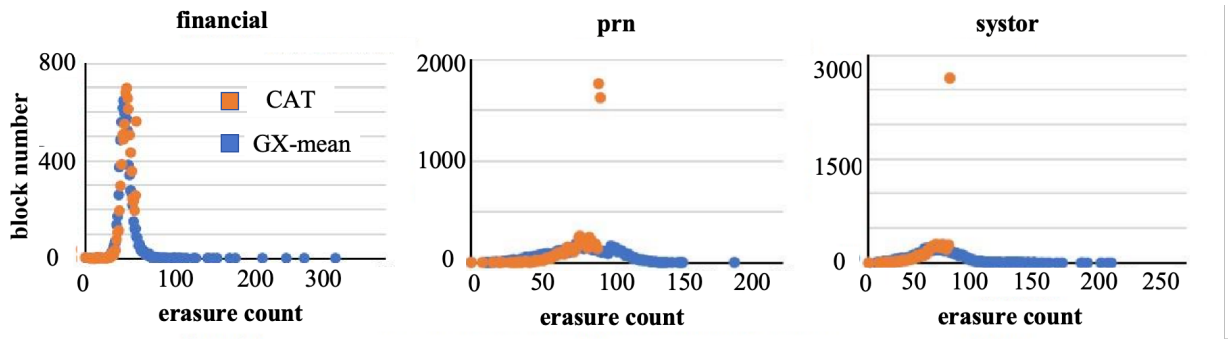
Before starting the experiment, the step two algorithm of X-mean GC must be fixed, because neither Greedy nor CB considers the erasure count information and they have no conflict with the step one of X-mean GC. We can



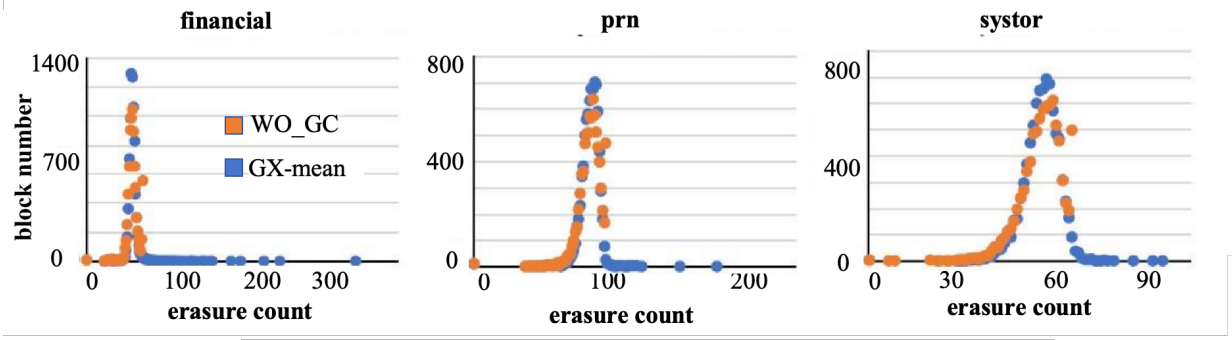
(a) Erasure count distribution of Greedy and GX-mean GC.



(b) Erasure count distribution of CB and GX-mean GC



(c) Erasure count distribution of CAT and GX-mean GC.



(d) Erasure count distribution of WO_GC and GX-mean GC

FIGURE 2. Erasure count distribution of different garbage collection schemes

TABLE V: Lifetime improvements of different algorithms

	GX-mean GC	GX-mean GC	CAX-mean GC	WX-mean GC
Financial	108.94%	99.44%	74.30%	52.74%
Pro	133.35%	47.78%	44.22%	6.83%
Systor	122.29%	76.03%	85.03%	6.54%

combine them with the step one of X-mean GC to form a complete algorithm. As CAT and WO_GC also take erasure count information into account, they can't be combined with X-mean GC directly. We extract the other two factors in CAT and WO_GC to guide the step two of X-mean GC. After combined with different algorithm, we called these newly formed X-mean GC: GX-mean GC, CX-mean GC, CAX-mean GC and WX-mean GC. Their algorithms of step two are shown in Table IV.

Figure 2 shows the erasure count distribution of different garbage collection schemes, the x-axis represents the erasure count of block and the y-axis represents how many blocks are holding the erasure count. The narrower the x-axis is, the more concentrated distribution of erasure count is and the better wear-leveling performance is. The diagrams in Figure 2 can be divided into four teams, Greedy vs GX-mean GC, CB vs CX-mean GC, CAT vs CAX-mean GC and WO_GC vs WX-mean GC. The blue points represent the original algorithms and the orange points represent the modified algorithms. In all these four teams, we can obtain the modified algorithms have obvious narrower x-axis. In the original algorithms, there are several blocks have extremely high erasure count which is very harmful to the stability of NAND flash-based storage system. After the intervention of X-mean GC, these unfavorable situations become much better.

We calculated the difference of the maximum erasure count (Max.N_{erase}) and the minimum erasure count (Min.N_{erase}), which is widely used to evaluate the performance of wear-leveling [27]-[29], [39]. From Figure 3, we can obtain that the modified algorithms can reduce the difference obviously and the least decline still reaches 30.8%.

In order to quantify the wear-leveling performance of X-mean GC, we also calculate the lifetime improvements of All the different algorithms when they are combined with X-mean GC. Lifetime improvements are computed according to the lifetime model and the results are listed in Table V.

From Table V, we find the improvements of lifetime are obvious in terms of GX-mean GC, CX-mean GC and CAX-mean GC. This is because Greedy, CB and CAT do not take erasure count information or the maximum erasure count into account, which is just the advantage of X-mean GC. For WX-mean GC, its original algorithm (WO_GC) also takes erasure count information and the maximum erasure count into account. It doesn't improve the lifetime as much as the other three. But it still shows a good performance on wear-leveling and achieves a big lifetime improvement for financial trace.

D. THE INFLUENCE OF THE X VALUE

In this section, we will explain why we set X value to 10. As we discussed in Section III-B, big X value will reduce the performance of wear-leveling and small X value will bring in more page copies. Two experiments were evaluated. The first one was used to study the relation between the lifetime improvement and the X value whereas the second one was used to study the relation between the number of pages copied out and the X value.

Figure 4 shows the relation between the lifetime improvement of WX-mean GC and X value, different colors mean different traces. In Figure 4, all three polylines of relative lifetime are approximately straight lines and their degree of linearity are 0.0080, 0.0035 and 0.0099 respectively. Therefore, we can regard that the wear-leveling performance and the X value have an opposite linear relationship.

Figure 5 shows the relation between the number of pages copied out of WX-mean GC and the X value. different colors mean different traces. All the three polylines do not show good linearity and this phenomenon is especially evident in the first half of these polylines.

In order to find the best balance point of performance and overhead, we calculate the relative change rate of the number of pages copied out. The expression of relative change rate represents the intensity of y's response to x's change. In Figure 6, the x-axis represents X value and the y-axis represents the relative change rate of the number of pages copied out. We find the relative change rate decreases with X value increasing for financial trace and prn trace, but the situation becomes different when it comes to systor trace. For systor trace, the relative change rate increases when the X value equals to 10, and it goes down as the other two polylines. That means the relative change rate is unstable when X value is smaller than 10. As we know, the wear-leveling performance declines with X value increasing. Therefore, the best balance point appears when X value equal to 10. We need to emphasize that 10 is really a small number comparing to the P/E cycles of NAND flash. It's just 0.4% of short-lived TLC's lifetime. Therefore, X-mean GC is able to ensure most memory cells to be worn out at the same time which is important for the stability of NAND flash-based storage system.

The reason why we select WX-mean GC as a reference is that the wear-leveling performance improvements of GX-mean GC, CX-mean GC and CAX-mean GC are much bigger than WX-mean GC, so they have more endurance on overhead.

E. OVERHEAD OF X-MEAN GC

In X-mean GC, a threshold is added to erasure count. As a result, more copy operation will be brought in when comparing to the original algorithm. We have reason to believe there is a growth of the total erasure count. In this section, we select the total erasure count to be the indicator of the overhead of X-mean GC.

Table VI shows the total erasure counts of different algorithms running under three different traces. The total

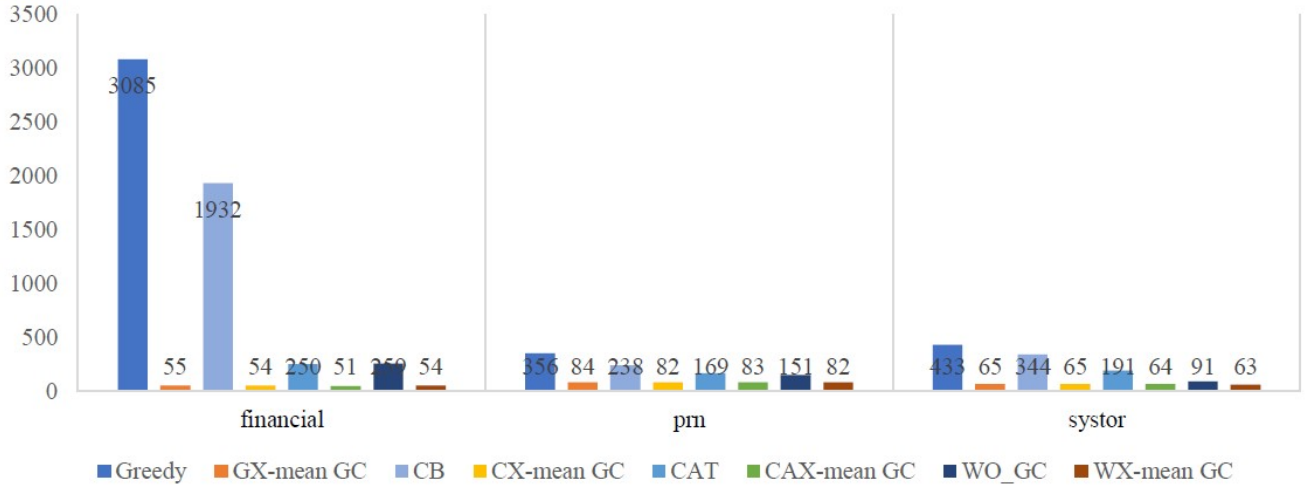


FIGURE 3. Difference of Max.Nerase and Min.Neraseand

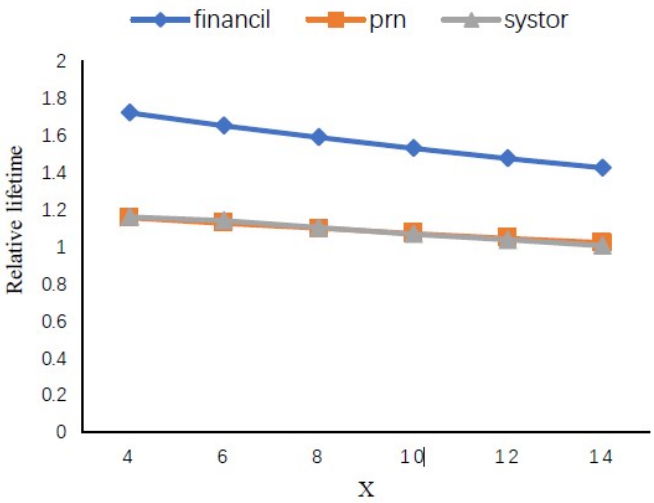


FIGURE 4. Relative lifetime comparing to original algorithms

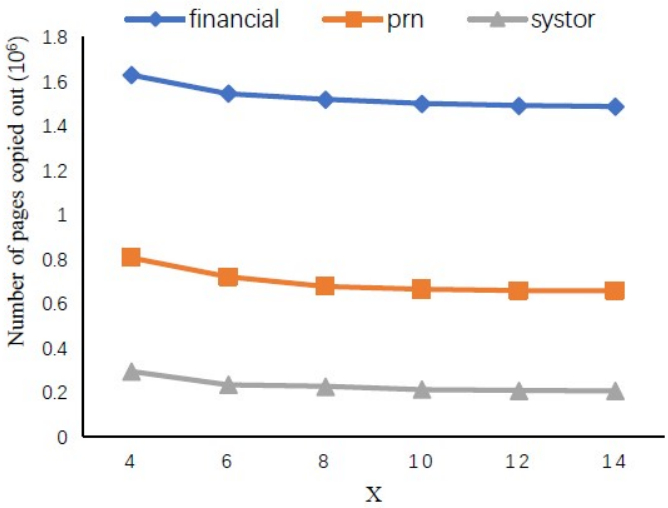


FIGURE 5. Number of pages copied out

TABLE VI: TOTAL ERASURE COUNT OF FAGC+

	Financial	Prn	Systor
12-1	169727	319305	92955
12-2	170163	318918	93559
12-3	170082	319772	92979
128-1	170265	319605	93477
128-2	174622	325468	96374
128-3	174627	325128	92643
256-1	174546	326260	96443
256-2	174717	324944	97965
256-3	174603	325468	96781

erasure count change rates between X-mean GC and original algorithms are also shown in Table VI.

From Table VI, we find that the total erasure count change rate is small. The biggest total erasure count change rate is 4.90% and this value will not cause big change in the performance of NAND flash-based storage system. However, these acceptable overheads contribute to big lifetime improvements.

Besides Greedy, CB, CAT and WO_GC, FaGC+ is also studied in this section. FaGC+ divides all pages into four kind according to the parameter “h”. Its purpose is to collect pages with similar access frequency to the same block and reduce overhead. It works good under Zipf distribution. However, thegarbage collection strategy doesn’t work well all the time. For example, if there is a page hasn’t been updated for a long

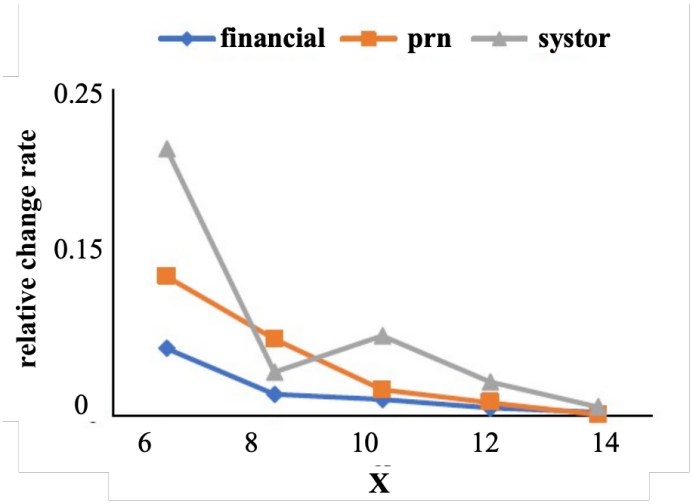


FIGURE 6. Relative change rate of pages copied out

TABLE VII: TOTAL ERASURE COUNTS AND CHANGE RATES OF DIFFERENT ALGORITHMS

(a) Total erasure count of Greedy and GX-mean GC

	Financial	Prn	Systor
Greedy	177505	281181	86324
GX-mean GC	177869	285610	88112
Change rate	0.21%	1.58%	2.07%

(b) Total erasure count of CB and CX-mean GC

	Financial	Prn	Systor
CB	171600	271988	84478
CX-mean GC	172314	272382	84894
Change rate	0.42%	0.14%	0.49%

(c) Total erasure count of CAT and CAX-mean GC

	Financial	Prn	Systor
CAT	145473	269332	70086
CAX-mean GC	151766	282530	72884
Change rate	4.33%	4.90%	3.99%

(d) Total erasure count of WO_GC and WX-mean GC

	Financial	Prn	Systor
WO_GC	150296	273534	69534
WX-mean GC	150621	273684	69648
Change rate	0.22%	0.05%	0.16%

time when it becomes invalid and of course its "d" becomes very big. Several this kind of pages may result in a big "CPS" even if there are a lot of valid pages in the same block. As a consequence, a large number of copy operations will be generated if this block is recycled. Because of the key roles "h" and "c" play in FaGC+, we conduct serval experiments with different "h" values and "c" values to study the total erasure count under different access patterns. In Table VII, the vertical axis title means different "h" values and "c" values. For example, 128-2 means "h" equals 128 and "c" equals 2.

From Table VII, we find the total erasure counts of FaGC+ have big differences comparing to other algorithms. The differences are relatively small under financial trace, but the differences are much bigger under the other two traces. FaGC+ doesn't shown any obvious advantage under the traces we use, even though it occupies extra storage space to store the relevant information of "h".

V. Discussion

In this section, we perform some discussions about X-mean GC, according to its design and evaluation results.

Application Scenarios. The proposed X-mean GC scheme demonstrates significant potential for deployment in NAND flash-based storage systems, particularly in environments where endurance and reliability are critical. Its ability to integrate seamlessly with existing garbage collection algorithms (e.g., Greedy, CB, CAT, WO GC) makes it applicable to a wide range of devices, including: (1) Consumer electronics: Smartphones, SSDs, and USB drives, where cost-effective wear-leveling is essential for prolonging device lifespan. (2) Enterprise storage systems: High-capacity SSDs in data centers, where uneven wear can lead to costly failures and downtime. Embedded Systems: Automotive and industrial IoT devices, which operate under harsh conditions and require robust memory management. (3) Cloud storage: Hybrid storage solutions combining NAND flash with DRAM or emerging non-volatile memory (NVM), where wear-leveling directly impacts system longevity and energy efficiency. The minimal overhead of X-mean GC (e.g., tracking average erasure counts) ensures compatibility with resource-constrained environments, such as edge computing devices.

Deployment. Deploying X-mean GC in real-world systems requires addressing several practical challenges: (1) Parameter tuning: while the paper sets X=10 based on simulations, real-world workloads (e.g., bursty writes, varying data locality) may necessitate adaptive adjustment of X. Future implementations could leverage runtime profiling to dynamically optimize X for specific use cases. (2) Hardware integration: the candidate pool mechanism relies on maintaining an up-to-date average erasure count. This can be implemented in firmware or embedded within NAND flash controllers with minimal additional hardware. (3) Compatibility with existing controllers: X-mean GC's two-step design allows integration with legacy systems. For instance, existing garbage collection algorithms (e.g., WO GC) can adopt X-mean as a pre-filtering step without overhauling their core logic. (4) Scalability: as NAND flash scales to higher densities (e.g., 3D NAND), the candidate pool's size must balance wear-leveling effectiveness and computational overhead. Techniques like hierarchical pooling or machine learning-based block classification could enhance scalability.

The proposed X-mean GC scheme represents a practical and flexible approach to wear-leveling, offering substantial endurance improvements with minimal overhead. Its

adaptability to existing algorithms and hardware makes it a strong candidate for near-term deployment in consumer and enterprise storage systems. Future research should focus on dynamic parameter optimization, cross-layer integration, and addressing emerging memory technologies to ensure X-mean GC remains relevant amid evolving storage landscapes.

VI. Conclusion

The endurance issue has been a persistent challenge since the inception of NAND flash memory. Traditional garbage collection schemes have struggled to impose strict limits on erasure counts, and it is difficult to accurately assess the wear state of each block. In contrast, we address these challenges by introducing a dynamic threshold based on the average erasure count of all blocks. This allows users to monitor the wear state of the NAND flash-based storage system through the average erasure count. Unlike traditional garbage collection schemes, the proposed X-mean GC consists of two independent steps, with step two being highly flexible and not bound by a rigid limit. This flexibility allows step two to be adjusted as needed. According to simulation results, the proposed scheme extends the lifetime of NAND flash-based storage systems by 133.35% with a reasonable overhead.

References

[1] M. Liu, L. Pan, and S. Liu, "Collaborative storage for tiered cloud and edge: A perspective of optimizing cost and latency," *IEEE Transactions on Mobile Computing*, 2024.

[2] Q. Xie, C. Zhang, and X. Jia, "Security-aware and efficient data deduplication for edge-assisted cloud storage systems," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 2191–2202, 2022.

[3] H. Hassan, A. Olgun, A. G. Yağlıkçı, H. Luo, O. Mutlu, and E. Zurich, "Self-managing dram: A low-cost framework for enabling autonomous and efficient dram maintenance operations," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 949–965.

[4] D.-H. Kim, H.-J. Kwon, and S.-J. Bae, "Dram circuit and process technology," in *Semiconductor Memories and Systems*. Elsevier, 2022, pp. 87–117.

[5] J. Jung, "Energy-efficient partial ldpc decoding for nand flash-based storage systems," *Electronics*, vol. 13, no. 7, p. 1392, 2024.

[6] K. Park, A. Lerner, S. Lee, P. Bonnet, Y. H. Song, P. Cudré-Mauroux, and J. Choi, "Babol: A software-defined nand flash controller," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1693–1705.

[7] S.-T. Lee and J.-H. Lee, "Review of neuromorphic computing based on nand flash memory," *Nanoscale Horizons*, 2024.

[8] H. Kim, S. Kim, J. Park, G. Byeon, and S. Hong, "Don't cache, speculate!: Speculative address translation for flash-based storage systems," *IEEE Access*, 2025.

[9] D. R. Purandare, S. Schmidt, and E. L. Miller, "Persimmon: an append-only zns-first filesystem," in *2023 IEEE 41st International Conference on Computer Design (ICCD)*. IEEE, 2023, pp. 308–315.

[10] M. Ye, Q. Li, Y. Lv, J. Zhang, T. Ren, D. Wen, T.-W. Kuo, and C. J. Xue, "Achieving near-zero read retry for 3d nand flash memory," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2024, pp. 55–70.

[11] S. Yang, X. Zhao, K. Xie, X. Zhan, J. Wu, and J. Chen, "One-shot read processing to enhance cold data retention in charge-trap tlc 3d nand flash," in *2023 IEEE 15th International Conference on ASIC (ASICON)*. IEEE, 2023, pp. 1–4.

[12] Y. M. Park, J. Yeom, D. Kim, and E.-Y. Chung, "Unified wear-leveling technique for nvm-based buffer of ssd," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[13] J. Wu, W. Li, L. Wu, M. Yuan, C. J. Xue, J. Xue, and Q. Li, "Effective stack wear leveling for nvm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 10, pp. 3250–3263, 2023.

[14] W. Xu and I. Koren, "A scalable wear leveling technique for phase change memory," *ACM Transactions on Storage*, vol. 20, no. 1, pp. 1–26, 2024.

[15] J. Zhang, C. Wang, Z. Zhu, D. Kline, A. K. Jones, H. Yang, and Y. Wang, "Realizing extreme endurance through fault-aware wear leveling and improved tolerance," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 964–976.

[16] Z. Yu, C. Yang, R. Zhang, P. Tian, X. He, L. Zhou, H. Li, and D. Liu, "Wear-leveling-aware buddy-like memory allocator for persistent memory file systems," *Future Generation Computer Systems*, vol. 150, pp. 37–48, 2024.

[17] Y. Song, Y. Lv, and L. Shi, "Adaptive differential wearing for read performance optimization on high-density nand flash memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[18] L. Long, S. He, J. Shen, R. Liu, Z. Tan, C. Gao, D. Liu, K. Zhong, and Y. Jiang, "Wa-zone: Wear-aware zone management optimization for lsm-tree on zns ssds," *ACM Transactions on Architecture and Code Optimization*, vol. 21, no. 1, pp. 1–23, 2024.

[19] R. Liu, C. Ran, H. Hu, A. Xiong, P. Chen, and L. Long, "Gap: A global wear-aware block pool for enhancing lifetime of zns ssds," in *2024 13th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2024, pp. 1–6.

[20] Y. Lv, L. Shi, Y. Song, and C. J. Xue, "Access characteristic guided partition for nand flash-based high-density ssds," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 4643–4656, 2023.

[21] Y. Wei, G. E. Blelloch, P. Fatourou, and E. Ruppert, "Practically and theoretically efficient garbage collection for multiversioning," in *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2023, pp. 66–78.

[22] 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 *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–6.

[23] O. Kwon, K. Koh, J. Lee, and H. Bahn, "Fegc: An efficient garbage collection scheme for flash memory based storage systems," *Journal of Systems and Software*, vol. 84, no. 9, pp. 1507–1523, 2011.

[24] C. Matsui, A. Arakawa, C. Sun, and K. Takeuchi, "Write order-based garbage collection scheme for an lba scrambler integrated ssd," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 510–519, 2016.

[25] S. Yu, N. Xiao, M. Deng, F. Liu, and W. Chen, "Redesign the memory allocator for non-volatile main memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–26, 2017.

[26] L. Han, Y. Ryu, and K. Yim, "Cata: A garbage collection scheme for flash memory file systems," in *International Conference on Ubiquitous Intelligence and Computing*. Springer, 2006, pp. 103–112.

[27] J. Hu, H. Jiang, L. Tian, and L. Xu, "Gc-arm: Garbage collection-aware ram management for flash based solid state drives," in *2012 IEEE Seventh International Conference on Networking, Architecture, and Storage*. IEEE, 2012, pp. 134–143.

[28] D. Stefanović, K. S. McKinley, and J. E. B. Moss, "Age-based garbage collection," in *Proceedings of the 14th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, 1999, pp. 370–381.

[29] M. Wu and W. Zwaenepoel, "envy: a non-volatile, main memory storage system," *ACM SIGOPS Operating Systems Review*, vol. 28, no. 5, pp. 86–97, 1994.

[30] A. Kawaguchi, S. Nishioka, and H. Motoda, "A flash-memory based file system." in *USENIX*, 1995, pp. 155–164.

[31] M.-L. Chiang, P. C. Lee, and R.-C. Chang, "Managing flash memory in personal communication devices," in *ISCE'97. Proceedings of 1997 IEEE International Symposium on Consumer Electronics (Cat. No. 97TH8348)*. IEEE, 1997, pp. 177–182.

[32] O. Kwon and K. Koh, "Swap-aware garbage collection for nand flash memory based embedded systems," in *7th IEEE International*

Conference on Computer and Information Technology (CIT 2007). IEEE, 2007, pp. 787–792.

[33] M. Lin, S. Chen, Y. Lu, and Z. Zhou, "Garbage collection policy for flash-aware linux swap system," *Electronics Letters*, vol. 47, no. 22, pp. 1218–1220, 2011.

[34] M. Lin and S. Y. Chen, "Swap time-aware garbage collection policy for nand flash-based swap system," *Electronics letters*, vol. 49, no. 24, pp. 1525–1526, 2013.

[35] J. Huang, Y. Hua, P. Zuo, W. Zhou, and F. Huang, "An efficient wear-level architecture using self-adaptive wear leveling," in *Proceedings of the 49th International Conference on Parallel Processing*, 2020, pp. 1–11.

[36] S. Yu, N. Xiao, M. Deng, Y. Xing, F. Liu, Z. Cai, and W. Chen, "Walloc: An efficient wear-aware allocator for non-volatile main memory," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.

[37] I. G. Kolokasis, G. Evdorou, S. Akram, C. Kozanitis, A. Papagiannis, F. S. Zakkak, P. Pratikakis, and A. Bilas, "Teraheap: Exploiting flash storage for mitigating dram pressure in managed big data frameworks," *ACM Transactions on Programming Languages and Systems*, vol. 46, no. 4, pp. 1–37, 2024.

[38] M. Nakanishi, Y. Adachi, C. Matsui, Y. Sugiyama, and K. Takeuchi, "Application-oriented wear-leveling optimization of 3d tsv-integrated storage class memory-based solid state drives," in *2018 International Conference on Electronics Packaging and iMAPS All Asia Conference (ICEP-IAAC)*. IEEE, 2018, pp. 27–32.

[39] H. Yan, Y. Huang, X. Zhou, and Y. Lei, "An efficient and non-time-sensitive file-aware garbage collection algorithm for nand flash-based consumer electronics," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 1, pp. 73–79, 2018.

[40] S. Mittal and J. S. Vetter, "A survey of architectural approaches for data compression in cache and main memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1524–1536, 2015.

[41] M. Qin, R. Mateescu, Q. Wang, C. Guyot, D. Vucinic, and Z. Bandic, "Garbage collection algorithms for meta data updates in nand flash," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–5.

[42] J. Guo, C. Min, T. Cai, and Y. Chen, "A design to reduce write amplification in object-based nand flash devices," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2016, pp. 1–10.

[43] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data retention in mlc nand flash memory: Characterization, optimization, and recovery," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 551–563.

[44] L.-P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, 2007, pp. 1126–1130.

[45] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, "Flashsim: A simulator for nand flash-based solid-state drives," in *2009 First International Conference on Advances in System Simulation*. IEEE, 2009, pp. 125–131.

[46] A. Gupta, Y. Kim, and B. Urgaonkar, "Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings," *Acm Sigplan Notices*, vol. 44, no. 3, pp. 229–240, 2009.