A Block Sequence Based Garbage Collection Scheme for NAND Flash Memory

Tuo Ding, Yuanze Liu*

***Abstract*—NAND flash memory is a mainstream storage technology widely used in modern devices, including personal computers and servers. It offers advantages such as shock resistance and high throughput. However, due to the limitations of erase/program cycles, NAND flash faces challenges such as limited lifespan, block wear, and uneven wear distribution. Additionally, garbage collection involves extensive erase operations, significantly impacting the endurance of flash memory. To mitigate block wear and enhance wear leveling, this paper proposes a block sequence-based garbage collection scheme (BS_GC) for NAND flash. The proposed approach introduces two key improvements: (1) an optimized block recycling policy and (2) a hot-cold data identification and separation mechanism using a block sequence table. Furthermore, the block sequence table is utilized to track block update frequency and erase counts. Compared to existing algorithms, BS_GC offers higher efficiency while requiring less RAM.**

***Index Terms*—Garbage collection, NAND flash memory, NAND flash controller**

I. Introduction

NAND flash-based solid-state drives (SSDs) have become one of the most popular storage media in laptops and personal computers, thanks to their excellent shock resistance, high processing throughput, and lightweight design. Furthermore, according to the IDC forecast in *Data Age 2025*, the share of flash memory in storage media is expected to grow significantly, driven by advancements in NAND flash technology [1].

Flash memory is a non-volatile, solid-state storage medium that includes both NOR and NAND flash. In NAND flash, 128 or 256 memory cells are connected in series along a bit line, with the bit and source lines multiplexed within the memory cells. This design reduces the memory cell area by half, lowering the cost per cell. More recently, the introduction of 3D NAND flash has significantly increased storage capacities while rapidly reducing costs [2].

In NAND flash, the erase operation is performed by releasing electrons through the bulk silicon electrode, erasing the entire block simultaneously. As a result, erasure must occur at the physical block level. However, a physical block consists of 64, 128, or 256 individual pages, which can be written independently. Notably, when data on a page is updated, the page itself cannot be erased separately.

Manuscript received November 22, 2024; revised April 12, 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).

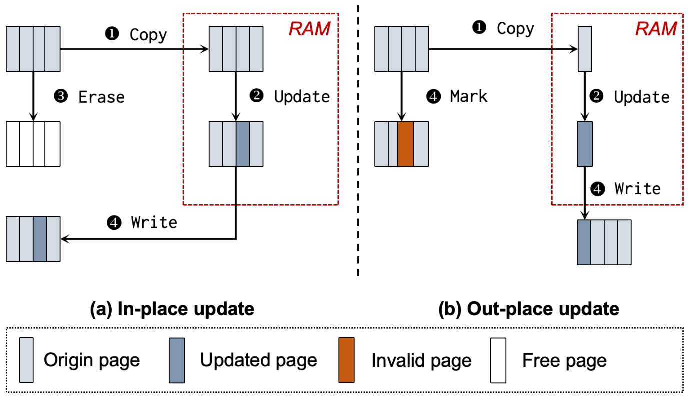


Fig.1. In place update and out place update

Therefore, updating data in NAND flash is inherently inefficient. With in-place updates, the controller must erase the entire block before writing the updated data back, leading to frequent block erasures, accelerated wear, and a reduced lifespan. In contrast, the out-of-place update method is more efficient. Instead of erasing and rewriting the original block, the controller writes updated data to new, free pages and marks the original data as invalid. Since this approach avoids immediate erase operations, it enhances efficiency and is widely adopted in NAND flash memory.

However, out-of-place updates lead to the accumulation of invalid pages, resulting in significant storage waste in NAND flash. Therefore, efficiently reclaiming these pages is essential. The flash controller manages this process through a garbage collection algorithm, which addresses three key considerations: how to recycle, which blocks to recycle, and when to recycle.

Traditional garbage collection algorithms consider three key factors when selecting blocks for recycling: the invalid page rate, the block erase count, and block age information. However, recording block age information efficiently remains a challenge, as it often requires significant RAM, especially given the large variability in block age.

The Cost-Benefit (CB) and Cost-Age-Times (CAT) algorithms use additional timers to track block age, but this information is lost in the event of a power failure [3], [4]. The Write Order-Based Garbage Collection (WO_GC) algorithm employs a Write Sequence Number (WSN) to track block age, but as the WSN grows over time, it demands substantial RAM [5].

In contrast, the proposed Block Sequence-Based Garbage

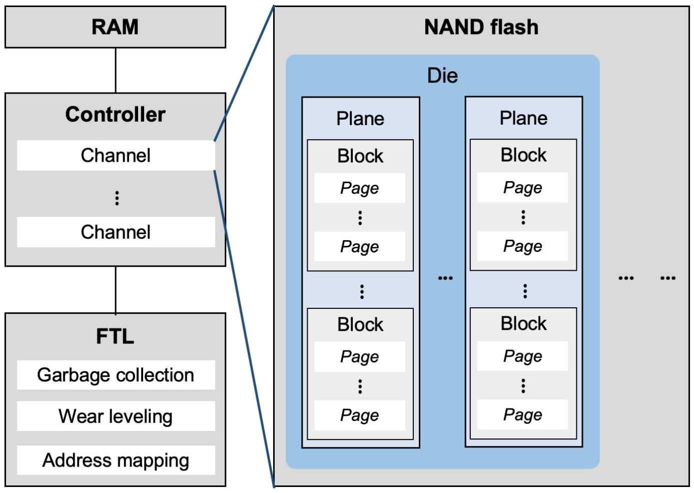


Fig. 2. The system architecture of NAND flash memory storage system.

Collection (BS_GC) algorithm records block age using the physical block number in the Block Sequence Table (BST), significantly reducing RAM usage. Additionally, BS_GC improves the block recycling policy, enhancing wear leveling performance.

NAND flash memory has a limited erase/program cycle. The maximum erase count per block is typically 100,000 for single-level cell (SLC) flash, 10,000 for multi-level cell (MLC) flash, and 1,000 for triple-level cell (TLC) flash. Therefore, ensuring even wear distribution across blocks is essential.

In garbage collection, identifying and separating cold and hot data helps reduce block wear and improve wear leveling. The proposed Block Sequence-Based Garbage Collection (BS_GC) algorithm leverages the Block Sequence Table (BST) to classify cold and hot data, storing them in separate blocks. This approach enhances wear leveling and significantly extends the lifespan of NAND flash memory.

II. RELATED WORK

In order to solve the constraints of NAND flash, the flash controller (flash translation layer) is necessary for the NAND flash storage system. Fig.2 indicates the architecture of flash storage system. The FTL is the core components in the flash controller, including garbage collection algorithm, address mapping algorithm and wear leveling algorithm.

A. Address mapping

In the NAND flash memory, the data can only be sequential write. Therefore, in order to emulate the random write, the controller develops the address-mapping algorithm in which the physical address number is the sequential write address and the logical write address. The address-mapping algorithm completes the translation from the logical address number to the physical address number and it usually uses a mapping table to record the corresponding relation between the physical address and the logical address.

According to the address translation granularity, there are three major types of FTL, including page-level mapping FTL, block-level mapping FTL and hybrid FTL. The page-level mapping FTL scheme maps each logical page address to a physical page address. It is flexible and simple for flash memory to realize the random write operation. However, in the page-level mapping FTL, the mapping table need record lots of mapping relation that consumes a large amount of SRAM. In order to decrease the RAM cost, Aayush et al. [6] proposes a page-level mapping algorithm, DFTL, this algorithm caches page-level address mappings selectively instead of saving all mappings in the RAM. The block-level mapping FTL only maps each logical block address to a physical block address and it contains fewer mappings. However, the block-level mapping FTL need extra operations to complete a page-write operation [7]. The hybrid FTL is developed to combine the advantages of the page-level mapping FTL and the block-level mapping FTL. The hybrid FTL contains both page-level mappings and block-level mappings. In the hybrid FTL, the data block that saves the newly writing data is mapped by the block-level mappings and the log block that saves the updated data is mapped by the page-level mappings. When garbage collection occurs, the controller needs merge the victim log block and data block. Lee et al. [8] propose a novel hybrid proposal, which is called FAST scheme. In FAST, the updated logical pages can be placed in any log block.

Table 1. Summary of wear leveling algorithm

Algorithm	Triggering condition	Wear Leveling policy
HC	Periodical	Swapping the data in the oldest block and in the youngest block
DP	Threshold	Partition blocks into hot pool and cold pool. The blocks in hot pool are for hot data and the blocks in cold pool are for cold data.
BET	Threshold	A bit erase table is used to count the block erase cycle and the algorithm uses the table identify the cold-hot block.
SD	Periodical	Static: hot-cold swapping. Dynamic: select the new block for write operation from a round-robin queue.
RRWL	Threshold	RRWL consistently uses one-to-one mode based on round robin method to increase the accuracy of cold block identification, with reduced memory size of a block erase table.

B. Wear leveling

In NAND flash memory, especially in the TLC NAND flash, the erase/program cycle is limited to the small. Therefore, in order to longer the lifetime and decrease the bad block, it is expected to wear all blocks out evenly. There are two types of wear leveling algorithm, including static wear leveling and dynamic wear leveling. The dynamic wear leveling scheme considers to select the blocks with the least erase count as the superior block for the next write operations. The static wear leveling considers to separate the cold-hot data and move the cold data into the block that has been erased more number of times. Table.1 shows some related work about wear leveling scheme. The hot-cold swapping (HC) algorithm considers swapping the data in the oldest block and in the youngest block if the erase count difference between the oldest block and the youngest block is too large [9]. The dual-pool (DP) algorithm considers using cold pool and hot pool to distinguish the old blocks and young blocks [10]. The BET algorithm uses a bit erase table (BET) to identify the cold-hot block [11]. The static-dynamic (SD) algorithm includes two parts, the static wear leveling scheme and the dynamic wear leveling scheme. The static wear leveling scheme considers swapping the data in the oldest block and in the youngest block and the dynamic wear leveling scheme considers selecting the superior block for write operation form a round-robin queue. The round-robin based wear leveling (RRWL) algorithm uses one-to-one mode based on round robin method to increase the accuracy of clod block identification [12].

C. *Garbage collection*

In the recent years, many considerable researches have focused on developing an efficient garbage collection scheme. Usually, there are four steps to complete garbage collection as following:

1. Judge whether the trigger condition has been reached.
2. Select the victim blocks to recycle.
3. Copy out the valid pages in the victim block into free pages.
4. Erase the victim block and update the free block list.

The trigger condition can be the threshold trigger or the regular trigger. In the threshold trigger, the garbage collection algorithm should set up a threshold value based on some parameters, such as the free block rate, update count or erase count. If the threshold value reaches, the garbage collection will be triggered. In the regular trigger, the garbage collection will be triggered regularly.

What is more, the victim block selection has been focused on in the garbage collection. Firstly, the block that has less valid pages should be selected as the victim block. In 1994, Wu and Zwaenepoel propose a greedy garbage collection scheme [13]. In the greedy garbage collection scheme, the victim block selection is only based on the valid pages number per block and the algorithm chooses the block that has the least valid pages as the victim block.

Secondly, the block age should be considered into the recycling policy because the older block contains more data errors and it is likely to be the cold data block. In 1995, Kawaguchi et al. [14] develop the cost benefit (CB) garbage collection algorithm and this algorithm considers the block age and the invalid page rate into the recycling policy. Comparing to the greedy algorithm, the CB algorithm has good performance in the page copied out and the block erase count. In 2011, Kwon et al. [9] develop the fast and efficient garbage collection algorithm (FeGC) and this algorithm considers the sum of the invalid page age into the recycling policy. In addition, the FeGC algorithm classifies the hot and cold page by the latest update time and the expected update time. The FeGC has good performance in the wear leveling, the garbage collection time cost and the energy consumption.

Thirdly, the block erase count is usually considered into the recycling policy in order to improve the wear leveling. In 1997, Chiang et al. [15] develop the cost-age-times (CAT) garbage collection algorithm and this algorithm considers the invalid page rate, the erase count and the block age into the victim block selection. In addition, the CAT algorithm uses an age transform function to calculate the block age in order to avoid block age being too large to overemphasize block age. In addition, the CAT algorithm identifies and separates cold-hot data by the block age and the block erase count to improve the wear leveling. In fact, the CAT algorithm has good performance in the wear leveling and reducing erase operation. In 2017, Matsui et al. [16] develop the write order-based garbage collection (WO_GC) scheme and this algorithm considers the invalid page rate, the erase count and the block age into the victim block selection. In addition, the WO_GC algorithm uses the write sequence number to record the block age and it can solve the problem that the block age will be lost if SSD shut down.

III. GARBAGE COLLECTION SCHEME BASED ON RELATIVE WRITE ORDER

A. *BS_GC*

In this paper, we propose a Block Sequence-Based Garbage Collection (BS_GC) scheme. The BS_GC algorithm employs a Block Sequence Table (BST) to manage garbage collection efficiently. The BST consists of two main areas: the **data block area** and the **free block area**. Each of these areas is composed of multiple Block Sequence Units (BSUs), where each BSU corresponds to a physical block. A BSU stores the physical block address and the block's erase count.

In the **data block area**, the **Block Sequence Number (BSN)** represents the position of a BSU within the BST, indicating the relative age of the corresponding physical block. A higher BSN value means that the BSU is positioned closer to the tail of the data block area, signifying a newer block. When a new block is selected for the next write operation, its corresponding BSU is placed at the tail of the data block area in the BST.

In the **free block area**, the BSUs are arranged based on their **block erase count**. BSUs with a lower erase count are placed at the head of the free block area, while those with a higher erase count are positioned toward the bottom. This arrangement helps in distributing wear evenly across blocks.

The **BS_GC algorithm** determines the victim block for garbage collection by considering three key factors: invalid

Table 2. Summary of garbage collection scheme

GC algorithm	Block recycling policy
Greedy	Choose the block with lowest score $score = u$
CB	Choose the block with highest score $score = \frac{age(1-u)}{u}$
CAT	Choose the block with lowest score $score = \frac{u}{1-u} \times \frac{1}{f(age)} \times N_{erase}$
FeGC	Choose the block with highest score $score = \sum_{i=1}^{n} age_i$
WO_GC	Choose the block with lowest score $score = \frac{u}{1-u} \times \frac{1}{\frac{Max.WSN - WSN}{Max.WSN}} \times \frac{N_{erase}}{Max.N_{erase}}$

u: percentage of valid page. N_{erase} : erase count.

age: time since last data updated.

WSN: write sequence number.

page rate, block erase count, and block sequence number. A block with a higher invalid page rate, a lower erase count, and a smaller BSN is more likely to be selected as the victim block. The selection process is guided by a specific scoring equation (Equation 2), where the block with the lowest score is chosen as the victim block for garbage collection.

$$\text{Score2} = \frac{u}{1-u} \times \frac{1}{\frac{blknum-BSN}{blknum}} \times \frac{N_{erase}}{Max.N_{erase}-N_{erase}} \qquad (2)$$

$$\frac{\text{dScore2}}{dN_{erase}} = \frac{u}{1-u} \times \frac{1}{\frac{blknum-RWSN}{blknum}} \times \frac{1}{(Max.N_{erase}-N_{erase})^2} \qquad (3)$$

The BS_GC algorithm offers two key advantages. First, it records block age information using the physical block number in the BST. Compared to the WO_GC algorithm, the physical block number requires fewer bits than the WSN, leading to lower RAM consumption. Second, according to Equation (3), the influence of the erase count in victim block selection increases as the erase count rises. This enhances the BS_GC algorithm's effectiveness in wear leveling.

B. Cold-hot data identification and separation

In the proposed BS_GC algorithm, cold-hot data identification and separation are essential for improving wear leveling. This process should follow the regulations outlined below:

(a) **Data Classification:** Initially written data is considered hot data. During garbage collection, if the BSU storing the victim block information is in the upper half of the data block area in the BST, the collected data is classified as cold, as the block has not been updated for a long time. Conversely, if the BSU is in the lower half of the data block area, the collected data is classified as hot.

(b) **Data Placement:** Hot data should be written into hot blocks, and cold data into cold blocks. When selecting a hot block for the next write operation, the algorithm chooses the block based on the BSU at the head of the free block area. If a cold block is selected, the algorithm picks the block according to the BSU at the bottom of the free block area.

(c) **Wear Leveling Optimization:** To enhance wear leveling, the algorithm periodically recycles cold data blocks stored in the BSU at the head of the data block area.

By adhering to these regulations, the algorithm can effectively separate cold and hot data during garbage collection, thereby improving wear leveling.

C. BST updating

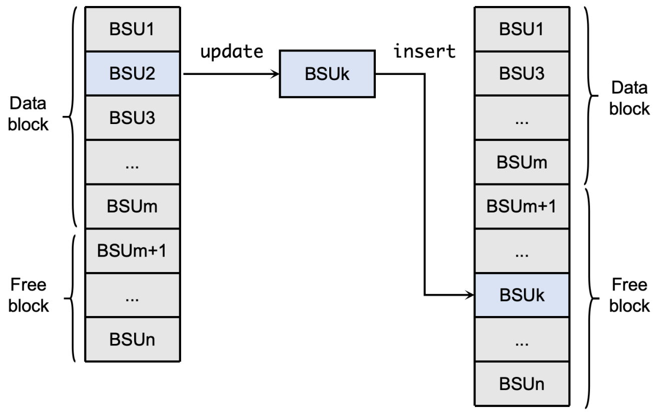


Fig.3. Update relative write order table after erase operation

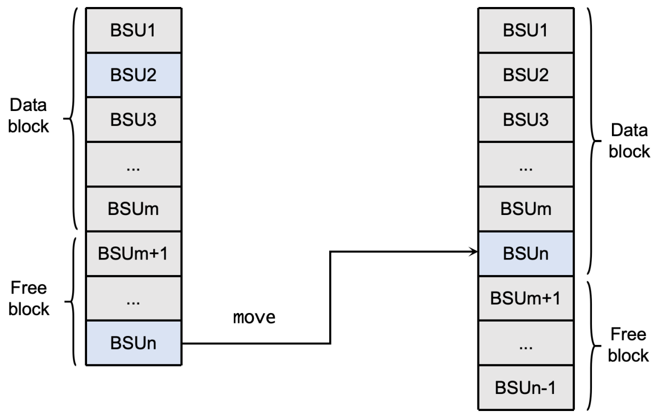


Fig.4. Update relative write order table after getting a new block for write operation

In the BS_GC algorithm, the BST must adhere to the following regulations:

Table 3. Flash memory information

Flash memory information	
Capacity	16Gb
Block size	256KB
page # per block	64
page size	4KB
Read latency	25us/page
Write latency	200us/page
Erase latency	2ms/block

(a) **Data Block Ordering:** The BSU storing data block information must be ordered by block update time. When a block is updated, the corresponding BSU should be moved to the tail of the data block area in the BST.

(b) **Free Block Ordering:** The BSU storing free block information must be ordered by the block's erase count. When a block is erased, the corresponding BSU should be inserted into the free block area based on the erase count.

To ensure that the BST remains effective for calculating the age of each physical block, the BS_GC algorithm must update the BST after obtaining a new block for a write operation and after erasing a physical block.

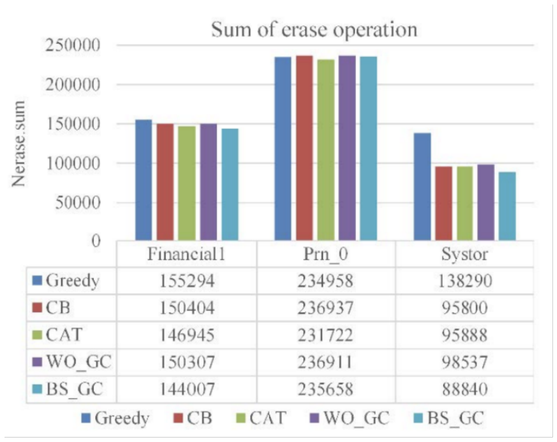
When a physical block is erased, the BST must update the BSU that stores the block's information. As shown in Fig. 4, the BSU (BSU2) that holds the erased block information must be updated. First, the erase count in BSU2 is incremented by one and updated into BSUk. Then, BSUk is removed from the data block area of the BST. Finally, the algorithm inserts BSUk into the free block area according to its updated erase count.

Furthermore, after obtaining a new block for a write operation, the BST needs to be updated. As illustrated in Fig. 5, if a cold block is selected, the algorithm moves the BSU (BSUn) that holds the cold block's information from the bottom of the free block area to the bottom of the data block area. If a hot block is selected, the algorithm simply shifts the boundary between the data block area and the free block area downward by one.

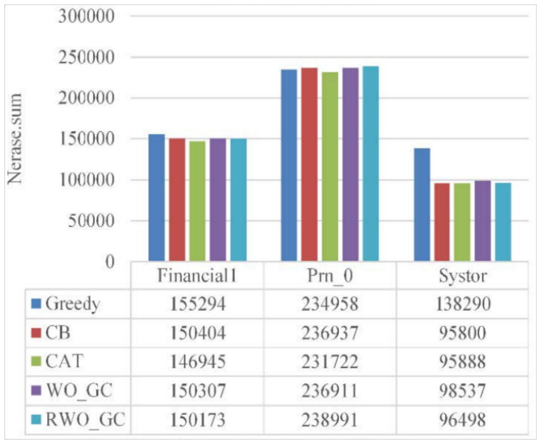
IV. EXPERIMENT

A. Simulation

In this section, we evaluate the proposed BS_GC algorithm using a flash simulator. To demonstrate its advantages, we compare BS_GC with traditional garbage collection algorithms, including the greedy algorithm, the Cost-Benefit (CB) algorithm, the Cost-Age-Times (CAT) algorithm, and the Write Order-Based Garbage Collection (WO_GC) algorithm. The flash simulator is configured based on the characteristics of the Micron MT29F16G08ADACA chip [16], with a total flash capacity of 16Gb. The details of the chip is shown in Table 3. In the simulation experiment, the input is the log-based trace including the financial1, prn_0, and systor traces. Our evaluation focuses on key performance metrics, including the total number of erase operations, the number of pages copied, the average garbage collection time, the energy consumption of garbage collection, the standard deviation of erase count distribution, and the average erase count distribution. Each experiment is conducted twice to illustrate the reliability of the results.



	Financial1	Prn_0	Systor
Greedy	155294	234958	138290
CB	150404	236937	95800
CAT	146945	231722	95888
WO_GC	150307	236911	98537
BS_GC	144007	235658	88840

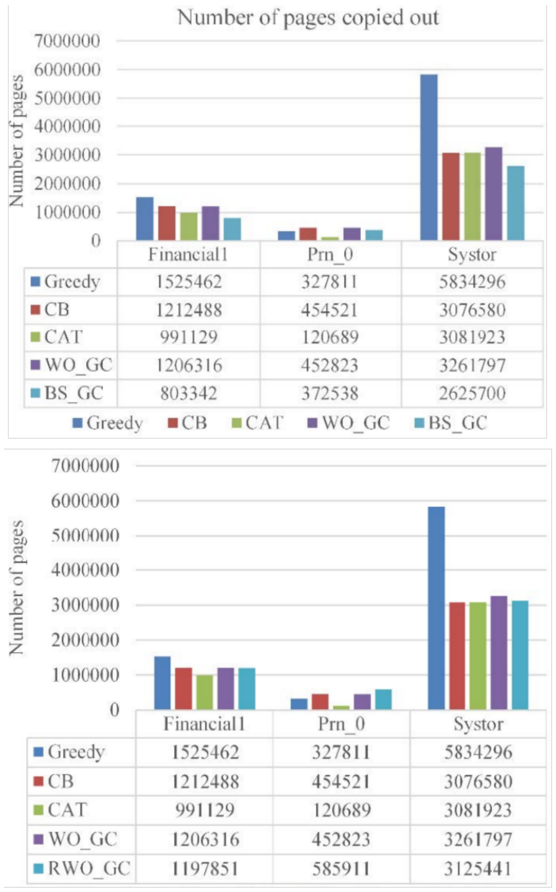


	Financial1	Prn_0	Systor
Greedy	155294	234958	138290
CB	150404	236937	95800
CAT	146945	231722	95888
WO_GC	150307	236911	98537
RWO_GC	150173	238991	96498

Fig.5. Sum of erase operation with different algorithm

B. Result analysis

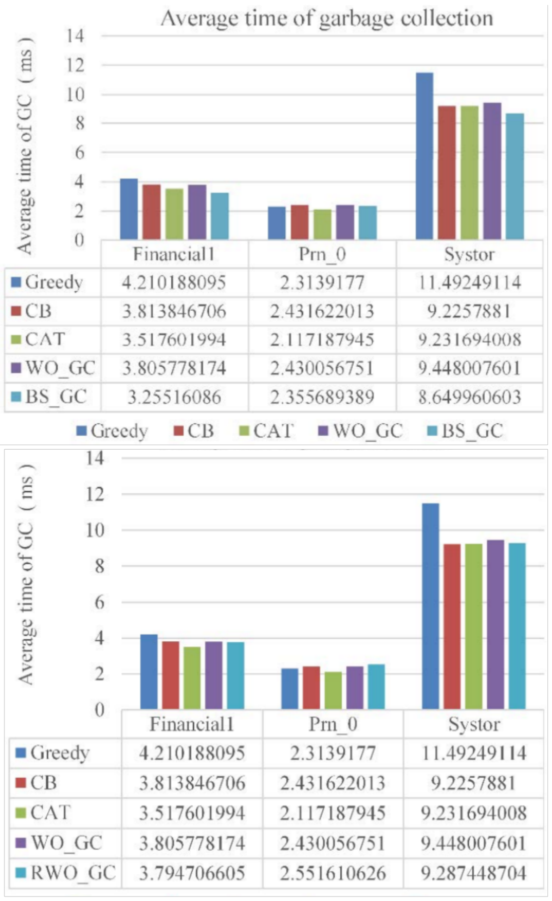
Figure 5 shows the simulation results for the total number of erase operations, which serve as an indicator of flash wear. As shown in Figure 5, compared to traditional algorithms, the proposed BS_GC algorithm reduces the number of erase operations by 2% in the *Financial1* workload and by 7.3% in the *Systor* workload. For this evaluation, the two sets of data in the experimental figure show a high degree of consistency. Both graphs depict the sum of erase operations for three datasets under different algorithms. The numerical values in the tables below the graphs remain largely the same across



	Financial1	Prn_0	Systor
Greedy	1525462	327811	5834296
CB	1212488	454521	3076580
CAT	991129	120689	3081923
WO_GC	1206316	452823	3261797
BS_GC	803342	372538	2625700

	Financial1	Prn_0	Systor
Greedy	1525462	327811	5834296
CB	1212488	454521	3076580
CAT	991129	120689	3081923
WO_GC	1206316	452823	3261797
RWO_GC	1197851	585911	3125441

Fig.6. Number of pages copied out with different algorithms



	Financial1	Prn_0	Systor
Greedy	4.210188095	2.3139177	11.49249114
CB	3.813846706	2.431622013	9.2257881
CAT	3.517601994	2.117187945	9.231694008
WO_GC	3.805778174	2.430056751	9.448007601
BS_GC	3.25516086	2.355689389	8.649960603

	Financial1	Prn_0	Systor
Greedy	4.210188095	2.3139177	11.49249114
CB	3.813846706	2.431622013	9.2257881
CAT	3.517601994	2.117187945	9.231694008
WO_GC	3.805778174	2.430056751	9.448007601
RWO_GC	3.794706605	2.551610626	9.287448704

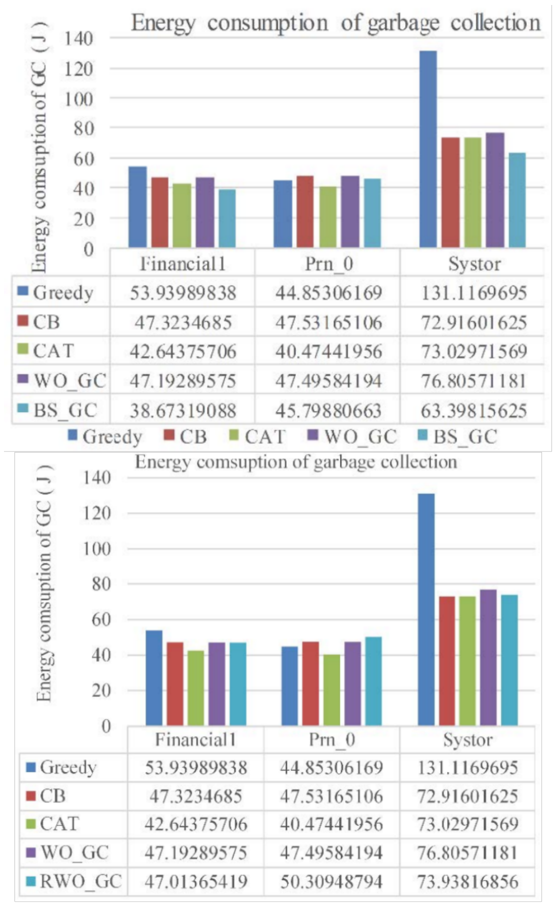
Fig.7. Latency of garbage collection with different algorithms

both figures for the respective algorithms. In Financial1, the erase operation counts remain similar across both graphs for each algorithm. In prn_0, the values in the second data are almost identical to those in the first, with only minor variations for some algorithms. For Systor, the erase operation counts are nearly identical, except for slight changes in the RWO_GC and BS_GC algorithms. The primary difference between the two graphs is the replacement of BS_GC in the first graph with RWO_GC in the second graph. The numerical values for other algorithms remain unchanged, confirming that the overall trend and data consistency between the two graphs remain strong.

Figure 6 presents the results for the number of pages copied. As shown, the BS_GC algorithm outperforms traditional algorithms in both the *Financial1* and *Systor* traces. Compared to traditional approaches, the number of pages copied using the BS_GC algorithm is reduced by 19% in *Financial1* and 14.7% in *Systor*. Additionally, the results from Figures 5 and 6 confirm that the proposed BS_GC algorithm is highly efficient.

In Figure 6, the two groups of data illustrate the number of pages copied out for different algorithms across three datasets (Financial1, Prn_0, and Systor). Upon comparison, the numerical values in both graphs remain largely consistent, demonstrating the reliability of the data. In Financial1, the values for all algorithms remain unchanged between the two graphs, indicating consistency in results. In prn_0, most values are the same, except for a minor variation in the RWO_GC algorithm in the second graph, which replaces BS_GC from the first graph. Systor Dataset: The overall trends remain unchanged, with RWO_GC showing a slightly different result compared to BS_GC while other algorithms maintain their original values. Overall, the two graphs depict a high degree of similarity, with the primary difference being the substitution of the BS_GC algorithm in the first graph with RWO_GC in the second. This suggests an alternative algorithm evaluation while keeping the other data points consistent.

Figure 7 illustrates the average garbage collection time across different algorithms, which represents the average time required to recycle a block during garbage collection. As shown in Figure 7, compared to traditional algorithms, the BS_GC algorithm reduces the average garbage collection time by 7.5% in *Financial1* and 6.2% in *Systor*, demonstrating its efficiency in time cost. This improvement is attributed to the BS_GC algorithm's strong performance in reducing both the total number of erase operations and the number of pages copied. According to the above figures, less pages are copied out in the prn_0 trace than the financial1 trace and systor trace. It means that the data in prn_0 trace update more frequently than the data in financial1 or systor. In the RWO_GC algorithm, because the effect of erase count



	Financial1	Prn_0	Systor
Greedy	53.93989838	44.85306169	131.1169695
CB	47.3234685	47.53165106	72.91601625
CAT	42.64375706	40.47441956	73.02971569
WO_GC	47.19289575	47.49584194	76.80571181
BS_GC	38.67319088	45.79880663	63.39815625

	Financial1	Prn_0	Systor
Greedy	53.93989838	44.85306169	131.1169695
CB	47.3234685	47.53165106	72.91601625
CAT	42.64375706	40.47441956	73.02971569
WO_GC	47.19289575	47.49584194	76.80571181
RWO_GC	47.01365419	50.30948794	73.93816856

Fig.8. Energy consumption of garbage collection with different algorithm

in the victim selection increase, it is easier for controller to recycle the block that contains data that is updated less frequently, which is good for the storage system to release space. However, all of the data in the prn_0 trace is updated frequently. It results in that the RWO_GC algorithm has more pages copied out in the prn_0 trace but less page copied out in the financial1 trace and the systor trace.

The data in both evaluations remain largely consistent, showing only slight variations in some values. For Financial1 dataset, the values for all algorithms remain identical in both graphs, ensuring consistency in results. For prn_0 dataset, most values are unchanged, with a slight difference in the RWO_GC algorithm in the second graph, which replaces BS_GC from the first graph. For Systor dataset, the trends remain the same, with minor differences in the RWO_GC algorithm's values compared to BS_GC, while other algorithms retain their original values. Overall, the two graphs are highly similar, with the primary distinction being the substitution of BS_GC in the first graph with RWO_GC in the second. This suggests a comparison between different algorithms while keeping the majority of the data unchanged.

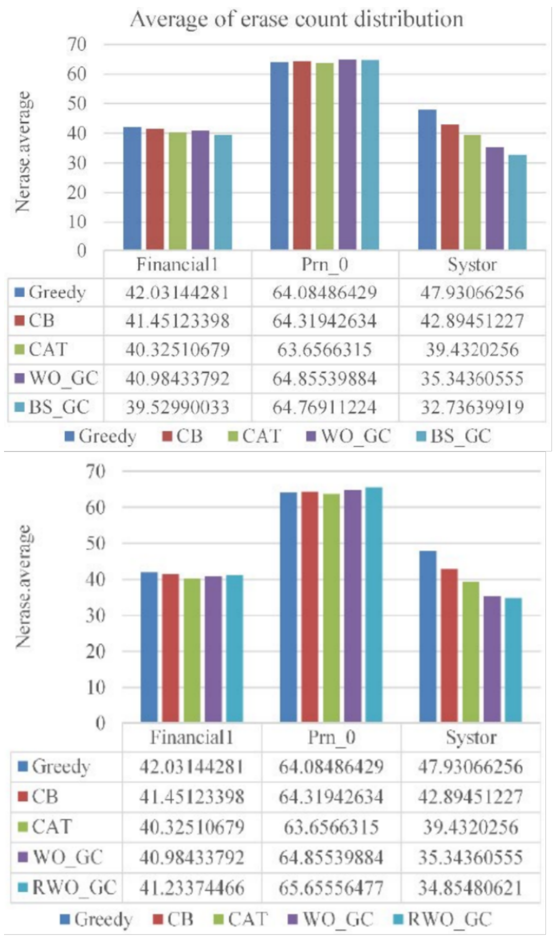
Figure 8 presents the simulation results for energy consumption across different algorithms, with a particular focus on energy usage during garbage collection. Usually, energy consumption is an important parameter in storage system. As shown in Figure 8, the BS_GC algorithm demonstrates lower energy consumption compared to traditional algorithms, primarily due to its efficient performance in reducing the total number of erase operations and the number of pages copied. In the prn_0 trace, the RWO_GC algorithm has the highest consumption because of the most pages copied out. However, in the financial1 trace, the RWO_GC algorithm has good performance due to its few erase operations and pages copied out. In addition, the energy consumption of RWO_GC algorithm is low in the systor trace.

The two sets of data presented in the image both depict the energy consumption of garbage collection (GC) across different systems or methods: Financial1, Prn_0, and Systor. The energy consumption values are measured for various GC strategies, including Greedy, CB, CAT, WO_GC, and BS_GC.

In both datasets, the energy consumption values for each GC strategy across the systems are consistent. For example, the Greedy strategy shows energy consumption values of 53.93989838 for Financial1, 44.85306169 for Prn_0, and 131.1169695 for Systor in both sets. Similarly, the CB strategy shows values of 47.32 for Financial1, 47.53 for Prn_0, and 72.91 for Systor in both datasets. The consistency extends to the other strategies as well, with minor variations in the second dataset introducing an additional strategy, RWO_GC, which shows values of 47.01365419 for Financial1, 50.30 for Prn_0, and 73.93 for Systor. This addition does not affect the consistency of the existing data points. Overall, the two datasets are highly consistent in their representation of energy consumption for the various garbage collection strategies across the different systems.

Figure 9 illustrates the average erase count distribution for different algorithms, which reflects block wear performance. The difference of average of erase count distribution between the algorithms is small in both of prn_0 trace and financial1 trace. According to Figure 9, compared to traditional algorithms, the BS_GC algorithm reduces the average erase count distribution by 2% in Financial1 and 6% in Systor. This improvement is attributed to the algorithm's effectiveness in minimizing erase operations. Additionally, the simulation results confirm that the proposed BS_GC algorithm significantly reduces block wear, enhancing the lifespan of NAND flash memory.

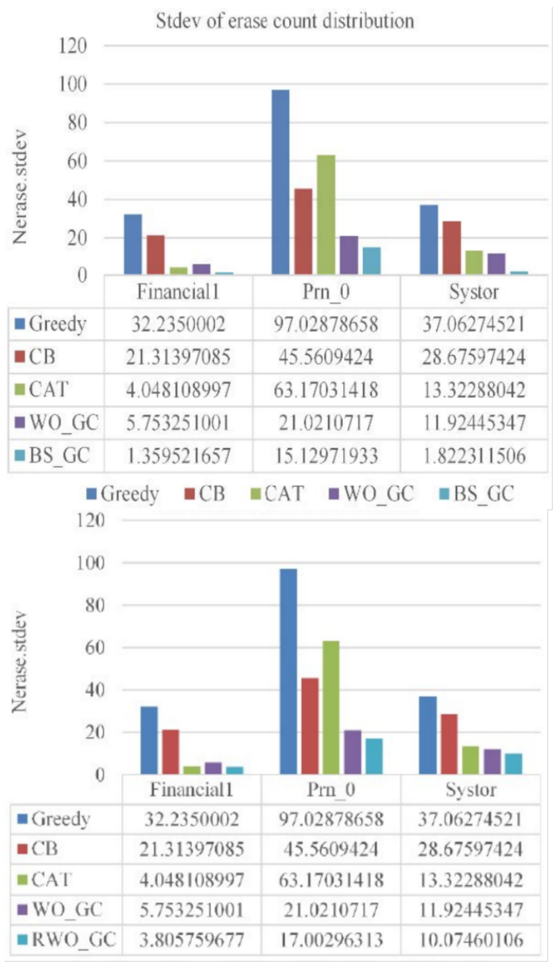
Both datasets display consistent values for the average erase counts across the systems. For instance, the Greedy strategy consistently reports averages of 42.03144281 for Financial1, 64.08486429 for prn_0, and 47.93066256 for Systor in both sets. Similarly, the CB strategy maintains values of 41.45123398 for Financial1, 64.31942634 for prn_0, and 42.89451227 for Systor across both datasets. The other strategies also show consistent results, with the second dataset adding the RWO_GC strategy, which records averages of 41.23374466 for Financial1, 65.65556477 for prn_0, and 34.85480621 for Systor. This addition does not disrupt the



	Financial1	Prn_0	Systor
Greedy	42.03144281	64.08486429	47.93066256
CB	41.45123398	64.31942634	42.89451227
CAT	40.32510679	63.6566315	39.4320256
WO_GC	40.98433792	64.85539884	35.34360555
BS_GC	39.52990033	64.76911224	32.73639919

	Financial1	Prn_0	Systor
Greedy	42.03144281	64.08486429	47.93066256
CB	41.45123398	64.31942634	42.89451227
CAT	40.32510679	63.6566315	39.4320256
WO_GC	40.98433792	64.85539884	35.34360555
RWO_GC	41.23374466	65.65556477	34.85480621

Fig.9. Average of erase count distribution



	Financial1	Prn_0	Systor
Greedy	32.2350002	97.02878658	37.06274521
CB	21.31397085	45.5609424	28.67597424
CAT	4.048108997	63.17031418	13.32288042
WO_GC	5.753251001	21.0210717	11.92445347
BS_GC	1.359521657	15.12971933	1.822311506

	Financial1	Prn_0	Systor
Greedy	32.2350002	97.02878658	37.06274521
CB	21.31397085	45.5609424	28.67597424
CAT	4.048108997	63.17031418	13.32288042
WO_GC	5.753251001	21.0210717	11.92445347
RWO_GC	3.805759677	17.00296313	10.07460106

Fig.10. Standard deviation of erase count distribution

consistency of the existing data. In summary, the datasets are consistent in their representation of the average erase counts for the evaluated GC strategies across the systems. The inclusion of the RWO_GC strategy in the second dataset provides further insights while maintaining the integrity of the original data.

Figure 10 presents the standard deviation of the erase count distribution across different algorithms. The standard deviation is strongly correlated with wear leveling performance, as a lower value indicates more uniform wear distribution across blocks.

According to Figure 10, the proposed BS_GC algorithm outperforms all other algorithms in minimizing the standard deviation of erase count distribution across all traces. In the *Financial1* trace, the BS_GC algorithm reduces the standard deviation by 28%. In the *Systor* trace, the reduction is even more significant, reaching 85%. In the financial1 trace, the proposed RWO_GC algorithm has the smallest standard deviation between these algorithms. It indicates that the RWO_GC algorithm has good performance in the wear leveling and the results is agreeable with the expected results. In the prn_0 and systor trace, the results are also agreeable with the expectation. In addition, comparing to the WO_GC algorithm, the standard deviation of erase count is 20% smaller in the prn_0 trace, 16% smaller in the systor trace and 33% smaller in the financial1 trace significantly. It indicates the improvement of the RWO_GC algorithm is meaningful in the wear leveling. These results align well with expectations and demonstrate that the BS_GC algorithm effectively improves wear leveling in NAND flash memory.

By analyzing the standard deviation, one can assess the reliability and stability of each GC strategy. A lower standard deviation suggests more consistent performance, while a higher value indicates greater variability. This information is essential for evaluating the effectiveness and predictability of different garbage collection methods in managing memory and storage resources.

The inclusion of this figure in the dataset underscores the importance of not only average performance metrics but also the consistency of those metrics, offering a more comprehensive view of the GC strategies' behavior across various systems.

Figure 11 illustrates the erase count distribution across different algorithms. In each diagram of Figure 12, the horizontal axis represents the erase count per block, while the

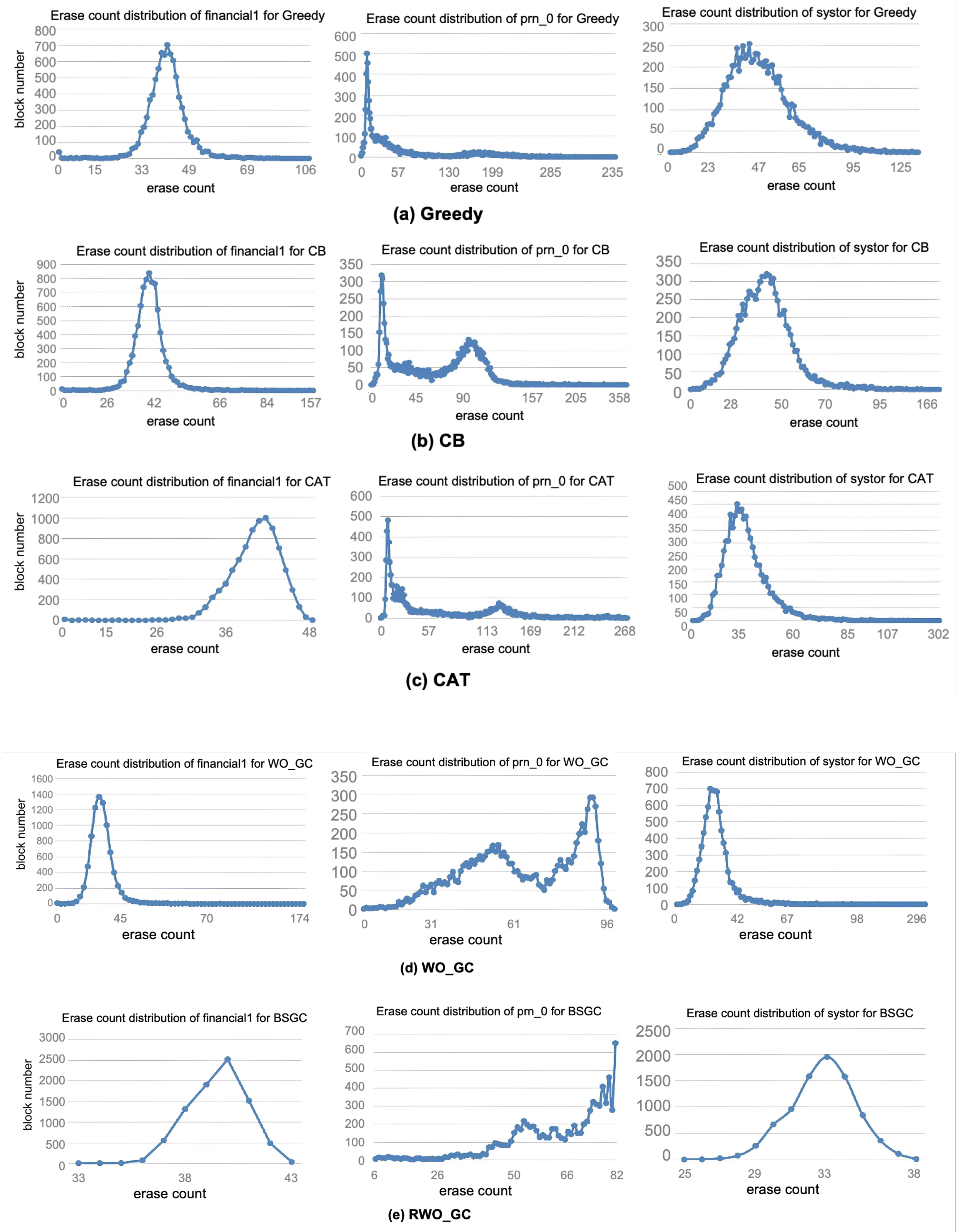


Fig.11. Erase count distribution of financial1, prn_0 and systor trace with different algorithm

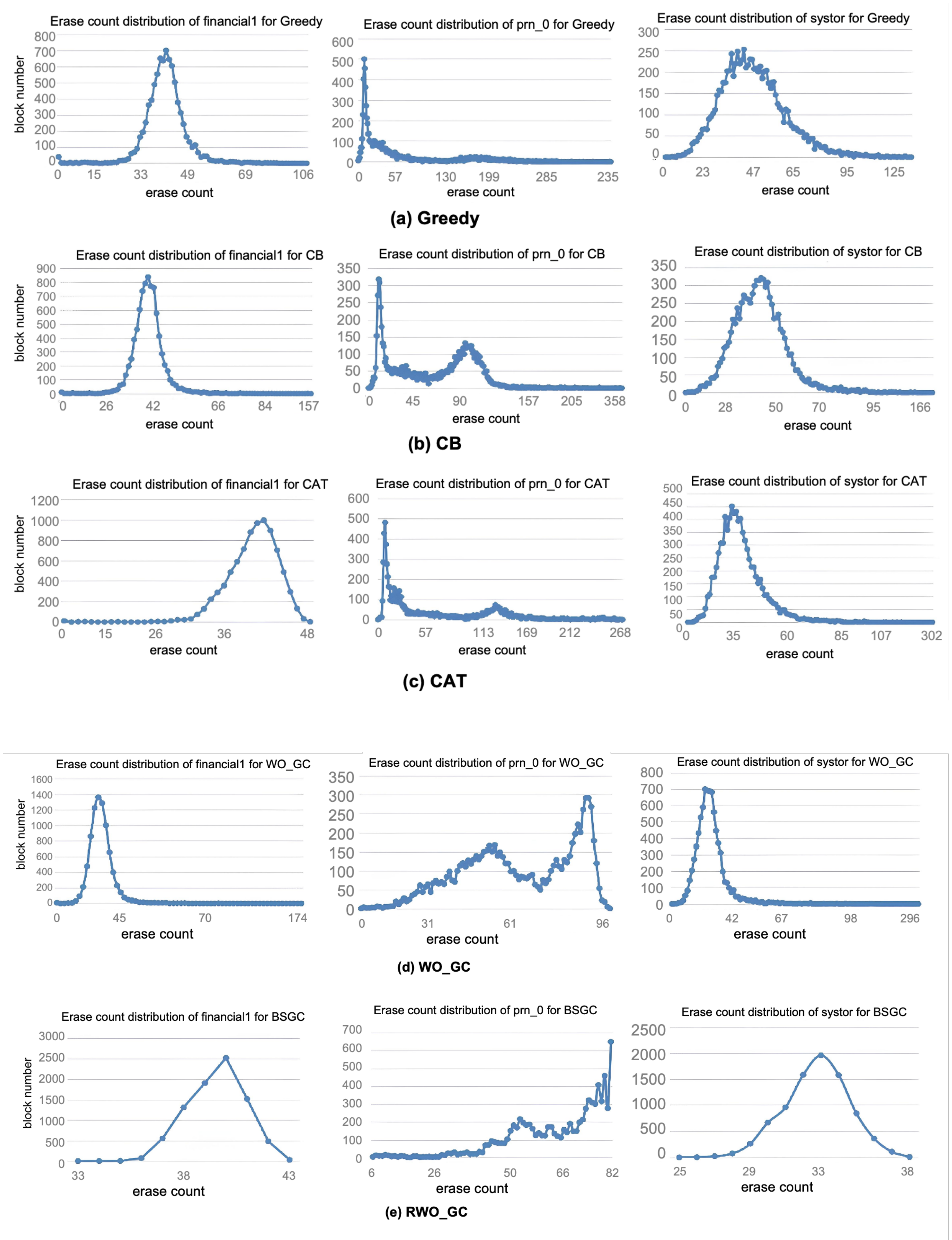


Fig.12 Erase count distribution of financial1, prn_0, and systor trace with different algorithm

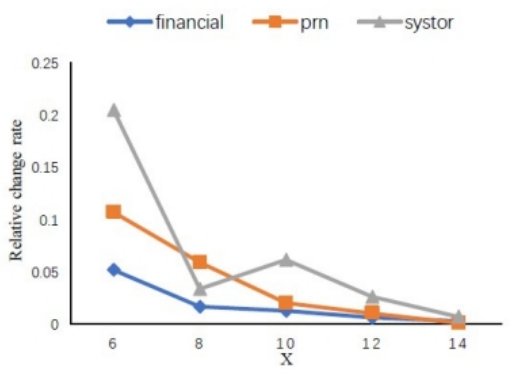


Fig.13. Relative change rate of pages copied out

vertical axis indicates the number of blocks corresponding to each erase count.

In Figure 11, the concentration of the erase count distribution reflects the effectiveness of wear leveling. The results show that, across all three traces, the erase count distribution in the BS_GC algorithm is more concentrated compared to other algorithms. This indicates that the BS_GC algorithm achieves better wear leveling performance, ensuring a more uniform distribution of erase operations across NAND flash blocks.

To determine the optimal balance between performance and overhead, we calculate the relative change rate of the number of pages copied out. This metric represents the sensitivity of the number of copied pages (y) to variations in a given parameter (x).

In Figure 13, the x-axis represents the X value, which is a critical parameter in the context of the experiment, while the y-axis represents the relative change rate of the number of pages copied out. This metric is essential for understanding how efficiently pages are being managed and copied within the storage system under different conditions. The graph compares the relative change rate across three different traces: financial1, prn_0, and systor, each representing distinct workload patterns.

For the financial and prn_0 traces, the relative change rate exhibits a clear downward trend as the X value increases. This indicates that as X grows, the system becomes more efficient in managing page copies, resulting in a reduced rate of change. This behavior suggests that higher X values lead to more stable and predictable performance in these traces, which is beneficial for applications requiring consistent storage performance.

However, the systor trace presents a more complex pattern. Initially, when the X value is below 10, the relative change rate is unstable, showing fluctuations. At X is 10, there is a noticeable increase in the relative change rate before it starts to follow a downward trend similar to the financial1 and prn_0 traces. This initial instability suggests that the systor trace, which likely represents a different type of workload, reacts differently to lower X values. The increase at X is 10 might indicate a threshold where the system's management strategy shifts, leading to a temporary spike in the change rate before stabilizing.

The optimal balance point for wear-leveling performance is observed at X is 10. This value represents a critical juncture where the system achieves a balance between efficient page management and wear-leveling effectiveness. Beyond this point, while the relative change rate continues to decrease, the wear-leveling performance generally declines, which could lead to uneven wear on memory cells over time.

It is important to note that X is 10 is a relatively small value when compared to the program/erase (P/E) cycle limits of NAND flash memory. Specifically, it represents only 0.4% of the total lifespan of short-lived TLC NAND. This highlights the efficiency of the BS_GC algorithm in ensuring that most memory cells wear out evenly, which is crucial for maintaining the long-term stability and reliability of NAND flash-based storage systems. By achieving even wear distribution, the algorithm helps to prevent premature failure of memory cells, thereby extending the overall lifespan of the storage device.

In summary, Figure 13 provides valuable insights into the dynamics of page management and wear-leveling under different workload conditions. The financial1 and prn_0 traces show a consistent improvement in efficiency with increasing X values, while the systor trace reveals a more nuanced behavior with an initial instability and a critical threshold at X is 10. The BS_GC algorithm's ability to maintain even wear distribution at such a low X value underscores its effectiveness in enhancing the durability and performance of NAND flash storage systems. This analysis not only highlights the importance of optimizing the X value but also emphasizes the need for tailored management strategies to handle diverse workload patterns effectively.

V. CONCLUSION

In NAND flash memory, garbage collection algorithm plays an important role in FTL. Garbage collection contains many erase operation and influences the performance of flash strongly. This work proposes a block sequence based garbage collection algorithm. The proposed BS_GC algorithm aims at improving the garbage collection efficiency and the wear leveling.

In addition, the BS_GC is compared with other garbage collection algorithm by the simulation experiment. According to the simulation results, the BS_GC algorithm has good performance in wear leveling, time cost, energy consumption, pages copied out and endurance. The sun of erase operations with BSGC algorithm decreases by 2% in the Financial1 and 7.3% in the Systor. The number of page copied out with BSGC algorithm decreases by 19% in the Financial1 and 14.7% in the Systor. The average of erase count distribution with BSGC algorithm decreases by 2% in the Financial1 and 6% in the Systor. The standard deviation of erase count distribution with BSGC algorithm decreases by 66% in the Financial1, 28% in the Prn_0 and 85% in the Systor. The average time cost decreases by 7.5% in the Financial1 trace and 6.2% in the Systor trace. The energy consumption of garbage collection decreases by 9.3% in the Financial1 and 13% in the Systor.

References

[1] M. Kim, M. Liu, L. R. Everson, and C. H. Kim, "An embedded nand flash-based compute-in-memory array demonstrated in a standard logic process," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 2, pp. 625–638, 2021.

[2] W. Liu, F. Wu, X. Chen, M. Zhang, Y. Wang, X. Lu, and C. Xie, "Characterization summary of performance, reliability, and threshold voltage distribution of 3d charge-trap nand flash memory," *ACM Transactions on Storage (TOS)*, vol. 18, no. 2, pp. 1–25, 2022.

[3] S. M. Rumble, A. Kejriwal, and J. Ousterhout, "Log-structured memory for {DRAM-based} storage," in *12th USENIX Conference on File and Storage Technologies (FAST 14)*, 2014, pp. 1–16.

[4] M.-L. Chiang and R.-C. Chang, "Cleaning policies in mobile computers using flash memory," *Journal of Systems and Software*, vol. 48, no. 3, pp. 213–231, 1999.

[5] P. Desnoyers, "Analytic models of ssd write performance," *ACM Transactions on Storage (TOS)*, vol. 10, no. 2, pp. 1–25, 2014.

[6] 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.

[7] Z. Xu, R. Li, and C.-Z. Xu, "Cast: A page-level ftl with compact address mapping and parallel data blocks," in *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2012, pp. 142–151.

[8] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 3, pp. 18–es, 2007.

[9] 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.

[10] 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.

[11] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, "Improving flash wear-leveling by proactively moving static data," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 53–65, 2009.

[12] S. H. Kim, J. H. Choi, and J. W. Kwak, "Rrwl: Round robin-based wear leveling using block erase table for flash memory," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 5, pp. 1124–1127, 2017.

[13] 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.

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

[15] 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.

[16] C. Matsui, C. Sun, and K. Takeuchi, "Design of hybrid ssds with storage class memory and nand flash memory," *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1812–1821, 2017.