CoupledCB: Eliminating Wasted Pages in Copyback-based Garbage Collection for SSDs

Jun Li[1], Xiaofei Xu[2], Zhibing Sha[3], Xiaobai Chen[1], Jieming Yin[1]*, Jianwei Liao[3]
[1]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[2]School of Computing Technologies, RMIT University, Melbourne 3001, Australia
[3]College of Computer and Information Science, Southwest University, Chongqing 400715, China
Emails: {junli, chenxb86, jieming.yin}@njupt.edu.cn, xiaofei.xu@ieee.org, {zhibing.sha, liaotoad}@gmail.com

***Abstract*—The management of garbage collection poses significant challenges in high-density NAND flash-based SSDs. The introduction of the copyback command aims to expedite the migration of valid data. However, its odd/even constraint causes wasted pages during migrations, limiting the efficiency of garbage collection. Additionally, while full-sequence programming enhances write performance in high-density SSDs, it increases write granularity and exacerbates the issue of wasted pages. To address the problem of wasted pages, we propose a novel method called CoupledCB, which utilizes coupled blocks to fill up the wasted space in copyback-based garbage collection. By taking into account the access characteristics of the candidate coupled blocks and workloads, we develop a coupled block selection model assisted by logistic regression. Experimental results show that our proposal significantly enhances garbage collection efficiency and I/O performance compared to state-of-the-art schemes.**

***Index Terms*—SSDs; Garbage collection (GC); Copyback; Odd/even (OE) constraint; Coupled block.**

I. Introduction

High-density NAND flash-based SSDs are emerging as the mainstream storage solution due to advancements in cell density and 3D stacking technologies [1]. Unlike traditional hard disk drives, SSDs only support out-of-place updates, in which new data is flushed to a different flash page, rendering the original data page invalid [2]. To reclaim the storage occupied by invalid data, garbage collection (GC) is responsible for migrating valid data and resetting fully-invalid blocks [3]. As 3D stacking technology evolves, the number of pages within a block significantly increases, leading to a greater volume of valid data migration during GC in 3D NAND SSDs [4], [5], [6]. However, during GC operations, I/O requests aimed at the same channel cannot be serviced, generating adverse effects on the I/O responsiveness of incoming requests [7], [8]. To speed up the process of data migrations, the *copyback* command is introduced [9]. This command allows valid pages to migrate within the plane internally via the page register without involving the SSD controller, significantly reducing the migration overhead [10]. However, copyback has a limitation: the odd/even sequence of the source page must match that of the destination page, known as the *odd/even (OE) constraint* [11], [12]. Consequently, some pages will be wasted with invalid states, leading to degradations in GC efficiency and space utilization [13].

Furthermore, while 3D high-density SSDs significantly increase storage capacity, they also experience longer access latency. To address the reduced program throughput in high-density SSDs, full-sequence programming (FSP) technology [14], [15] has been introduced to enable writing a group of pages at once, resulting in a performance boost of up to $\mathtt{n}$ times [16]. However, this approach also increases the program granularity to $\mathtt{n}$ pages[1]. As a result, the OE constraint leads to more mismatches between GC victim and destination pages during copyback operations. To the best of our knowledge, no existing work comprehensively addresses the inefficiency issues of copyback in the context of FSP for SSDs.

To efficiently reduce wasted pages during copyback-based garbage collection in FSP-enabled SSDs, we propose a new method called *CoupleCB*. This method coordinates with other blocks in the garbage collection plane, called *coupled blocks*, to fill the mismatched pages induced by the OE constraint. Additionally, the filled pages from coupled blocks can be migrated in advance with minimal flash read overhead. In summary, this paper offers the following contributions:

- We propose a page migration method assisted by coupled blocks, which can utilize wasted space to proactively migrate pages in coupled blocks.
- We develop a coupled block selection model using the logistic regression (LR) approach, thus avoiding inefficient preemptive page movements.
- We evaluate our proposal using various disk traces from real-world applications. Compared to state-of-the-art methods, CoupleCB can reduce space wastage by 97.4% and improve I/O tail latency by 33.0% on average.

II. Background and Motivation

A. Background and Related Work

Garbage collection plays a crucial role in the flash translation layer (FTL). Its main purpose is to free up space taken by outdated data (referred to as invalid pages) that accumulates due to out-of-place updates, especially when the

*Corresponding author: Jieming Yin.

[1]$\mathtt{n}$ equals the cell density, and $\mathtt{n}$=3 in triple-level cell (TLC) SSDs. This paper focuses on TLC SSDs as they are widely used in commercial products

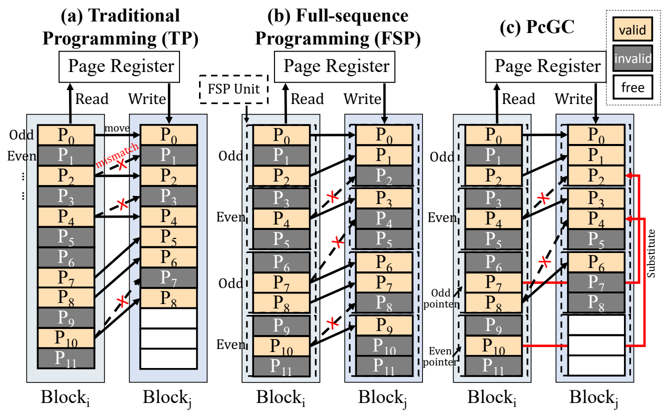

Fig. 1. An illustration of the traditional copyback in traditional programming (TP) [17] and FSP [16], and an advanced method PcGC [18]. Note that in TP of TLC SSDs, the program unit is page level, and the program sequence alternates between odd and even pages, similar to single-level cell (SLC) SSDs, because of minimizing the program disturbance of high-density SSDs [17], [19]. For illustration simplicity, the program sequence of TP is from 0 to the last page, but the odd and even sequence is alternated. In FSP, three adjacent pages within the dashed box constitute one FSP unit, sharing the odd/even sequence [11], [12].

available capacity of the SSD drops below a certain threshold. The GC process involves moving pages and performing erase operations. The conventional valid page movement policy, external data movement (EDM), involves reading valid data from the victim block and then writing them into the destination block via the SSD controller. An advanced command known as copyback (or internal data movement, IDM) was employed to expedite data movement during GC. This method allows data to be read and written directly via the plane page register, thereby saving time that would have been spent on bus transfers between the SSD controller and the page register.

While copyback can effectively accelerate data transmission during GC, it restricts how pages should be migrated. Specifically, the victim page and the destination page must have the same odd/even sequence, known as the *OE constraint* [13], [2]. For example, when migrating a valid page from a victim block to a free page in a free block, the addresses of the valid and the free page must be both odd or both even[2]. As a result, two consecutive odd (or even) victim valid pages must be migrated to two consecutive odd (or even) addresses in the new block, leading to a wasted page in between. As illustrated in Figure 1(a), during GC, data is migrated from $Block_i$ to $Block_j$ in an attempt to reclaim pages marked as invalid. In this scenario, p_1, p_3, and p_7 in $Block_j$ are wasted, because p_2, p_4, and p_{10} in $Block_i$ must follow the OE constraint.

Full-sequence programming, or FSP, can program multiple-bit information into a cell simultaneously, thereby dramatically enhancing the write throughput. For example in TLC SSDs, three pages can be programmed in a word line at a time. Prior work primarily focused on issues in FSP-enabled SSDs, such as read parallelism [20], sync write [16], and read variation [21]. However, no existing work has studied the inefficiency issue of copyback in the context of FSP. As illustrated in Figure 1(b), when the mandatory 3-page unit programming in the FSP scenario meets the OE constraint, three odd/even pages must always be aligned within the same FSP unit. As a result, there is a greater probability of mismatches and increased page wastage compared to when FSP is not used.

[2]In multi-level cell (MLC) SSDs, the odd/even sequence depends on their least significant bit (LSB) page [11], [12].

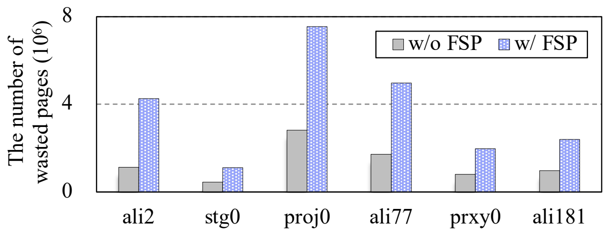

Fig. 2. The comparison of the wasted page number with/without FSP method.

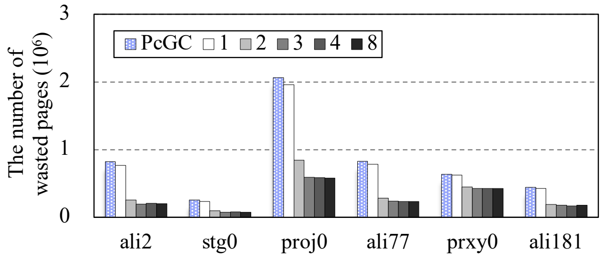

Fig. 3. The comparison of varied numbers of blocks assisting in copyback and the related work PcGC.

To address the OE constraint, Pang et al. proposed a page-state-aware cache (PSA-Cache) scheme [22]. This approach allows flushing valid data from the PSA-Cache residing in the internal DRAM, thereby preventing unnecessary page migration and reducing the number of wasted pages. Since most of the internal DRAM cache within SSDs is consumed by the page-level mapping table, the desired data may not be found in the cache.

Further, PcGC was introduced with two pointers pointing to the valid odd/even pages to match the victim page with the destination page [18]. This approach reduces wasted pages resulting from sequential migration by relocating valid pages from one FSP unit to another, as long as the two FSP units are both odd or both even, as shown in Figure 1(c). However, because the number of valid and invalid pages in odd/even FSP units might not always match, some pages are still wasted (e.g., p_5, p_7, and p_9 in $Block_j$ in Figure 1(c).). The issue of wasted pages caused by the OE constraint is still a significant challenge that needs to be addressed.

B. Motivational Experiments

In the first experiment, we compare the number of wasted pages with and without the FSP method to understand the impact of OE constraint in the context of FSP-enabled SSDs. The experiment settings and detailed specifications of selected traces are outlined in Section IV-A. As shown in Figure 2, the number of wasted pages increases by an average of 1.78× when implementing FSP technology. These results indicate the necessity of designing an efficient copyback scheme for the SSDs with FSP support.

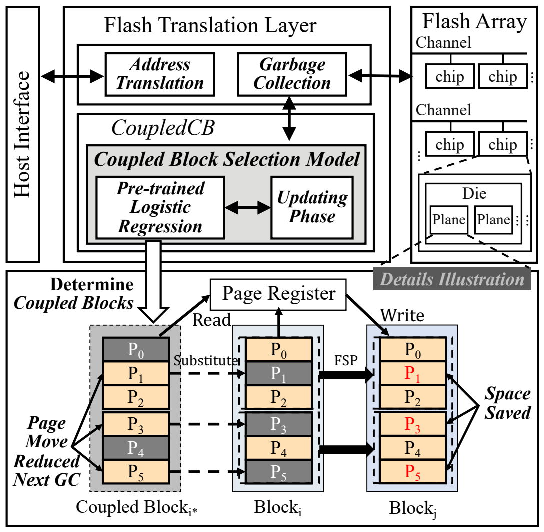

Fig. 4. The high level overview of the proposed CoupledCB with an illustration of copyback process guided by coupled block selection model.

When determining the GC block for data migration, SSDs commonly employ a greedy algorithm that goes through all blocks in the plane and selects the one with the most invalid pages in it [2], [23]. In the second experiment, we utilize multiple blocks as substitutes during page movements, with the goal of reducing the number of wasted pages. Specifically, along with the GC block identified by the baseline greedy algorithm, we retain one or multiple additional blocks (also with a high number of invalid pages), allowing valid pages from these additional blocks to fill up the "holes" from the GC block where OE constraint is not met. As shown in Figure 3, retaining one additional block (i.e.,1 in the figure) has a similar performance compared to PcGC. Including more additional blocks reduces wasted pages, but the improvement plateaus at 3 blocks for most applications. This result indicates that block selection significantly impacts GC efficiency, emphasizing the need for a more intelligent selection policy.

III. Design and Implementation of CoupledCB

A. Overview

Motivated by the experiments in Section II-B, the basic idea of CoupledCB is to select additional blocks (called *coupled blocks* in this work) within the GC victim plane, to fill up as many wasted pages caused by the OE constraint as possible. An overview of our design is shown in Figure 4. In the FTL, the garbage collection module selects the GC victim block, migrates valid data, and then erases the block to reclaim space. CoupledCB functions as a component of the garbage collection module, aiding in the migration of valid data. Specifically, the Coupled Block Selection Model determines the coupled blocks using logistic regression. When a GC victim block lacks odd/even pages to be migrated, CoupledCB is activated to find a suitable block for filling, thus reducing wasted space and allowing valid pages to be migrated in advance. As illustrated in Figure 4, p_1, p_3, and p_5 can be filled from *coupled block*$_{i*}$. Consequently, the wasted pages induced by the OE constraint can be significantly reduced. To achieve this, the coupled block must contain an adequate number of invalid pages that satisfy the OE constraint. These pages are not expected to be updated until the next garbage collection. The coupled block can then fill up the positions that would be wasted due to the OE constraint. Ultimately, our method can achieve excellent overall GC efficiency.

B. Benefit Analysis of CoupledCB

Through the second experiment in Section II-B, we found that the coupled block selection under the rule of greedy algorithm plateaus beyond three blocks. To improve coupled block selection so wasted pages can be further reduced, we first analyze the cost of CoupledCB. Since the write granularity of the FSP unit in TLC SSDs is 3 pages at a time, even if mismatches cause an FSP unit to contain invalid data, it still requires one FSP write (e.g., $p_0 - p_2$ in Figure 1). That is to say, the **cost** of the proposed CoupledCB mechanism only involves one additional read operation per migration from coupled blocks, without extra write operations, similar to the write time in the baseline method, as the write is incorporated into an individual FSP write. Since the invalid pages are filled up by CoupledCB, the **benefits** are the saved space that would be wasted by OE constraint, and one-page migration from coupled blocks in advance. Then, we can define the cost-benefit model as follows.

$$\begin{aligned} E &= (1 - P_{invalid}) \times (T_{space} + T_{read} + T_{write}) - T_{read} \\ &= (T_{space} + T_{write}) - P_{invalid} \times (T_{space} + T_{read} + T_{write}) \end{aligned} \tag{1}$$

where T_{space}, T_{read}, and T_{write} are the time induced by the saved space, read and write operation, respectively. $P_{invalid}$ indicates the probability of migrated pages from coupled blocks being invalid. In which, T_{read} and T_{write} are constant values. T_{space} cannot be directly measured since it is highly related to the system loads. Fortunately, T_{space} can be treated as an approximately constant value. Thus, E can be considered as a decreasing function of $P_{invalid}$. $\underline{\text{To obtain an optimal } E,}$ $\underline{\text{the problem can be transformed into minimizing } P_{invalid}}$. In other words, our best cost evaluation model can be achieved by selecting the filled pages that would not be updated during the next GC of their blocks. Then, we will describe how to estimate $P_{invalid}$ efficiently.

C. Coupled Block Selection Model

To estimate $P_{invalid}$ inside the garbage collection module efficiently yet effectively, we desire a machine learning algorithm that can be a) pre-trained offline, b) lightweight, and c) further tuned with new data.

Existing popular machine learning approaches share various limitations: (1) **Deep learning**: deep learning methods support pre-train before deployment and have shown promising performance in many tasks, such as data prefetching and cache replacement policy [24], [25], [26]. However, these methods may not well fit the specific workloads and could cause significant implementation and updating overhead in resource-limited SSDs. (2) **Reinforcement learning**: reinforcement learning is widely used in SSD-related optimizations, such as [27], [28], [29], [30]. Within the reinforcement learning framework, the agent learns from a reward signal designed by humans to reflect the desired objective. However, the reward function serves as a soft supervision signal [31], suffering from low sample efficiency.

Different from the deep learning and reinforcement learning approaches, logistic regression (LR) [32] is a classic machine learning method, with great sample efficiency and lightweight nature. Furthermore, LR can be trained offline before deployment, and subsequently tuned with new data, which greatly matches the requirements of our expected algorithm. Thus, we introduce logistic regression[3] to estimate $P_{invalid}$ for the candidate coupled blocks. Note that, $P_{invalid}$ is estimated at the block granularity rather than the page level, in order to limit search and space overhead.

The proposed LR-based coupled block selection model has three phases: pre-training, interacting, and updating.

1) Pre-training: The pre-training process is offline, which is coordinated with multiple real-world workloads to make the model learn a general pattern of the workloads. During training, the LR model is initialized as empty. To eliminate the impact of temporal order on training, the sequence of workloads in datasets will be shuffled per epoch.

Specifically, the proposed LR-based coupled block selection model is a supervised learning algorithm for binary classification that predicts the probability of a class using a sigmoid function. The objective of the model is to find a mapping function parameterized by θ, so that the output class probability can reflect the ground-truth class. The probability of the class is defined as:

$$P_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2}$$

where θ represents the parameters of the model and x represents the input feature vector. In which, the inputs in the context include the access features of block, plane and the whole SSD system. Specifically, we include the invalid ratio of coupled blocks and GC plane, Ir_B and Ir_P, as well as the write ratio and update ratio of SSD, Wr and Ur, as input features. Note that, We introduce these four parameters in both the current and previous time windows, totally 8 kinds, to reflect trends over time.

The goal of the LR model is to minimize the negative log-likelihood loss, i.e., the difference between the predicted and actual labels. The loss function is defined as:

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\log(P_\theta(x^{(i)})) + (1 - y^{(i)})\log(1 - P_\theta(x^{(i)}))\Big] \tag{3}$$

[3]Implemented based on https://github.com/atick-faisal/Logistic-Regression.

Algorithm 1: *Coupled block* selection in CoupledCB

```
Input: Parameters of candidate blocks
Output: Coupled blocks
1  load_model(LR);
2  /*Select coupled blocks if CoupledCB is activated.*/
3  while GC block searching do
4      /*Model checking during GC greedy algorithm*/
5      $P_{invalid} \leftarrow$ LR(parameters of $block_i$);
6      Random $\zeta$ between 0 to 1;
7      if $\zeta < P_{invalid}$ then
8          /*replace block with the least invalid pages;*/
9          add(coupled blocks, $block_i$);
10 /*Update LR model periodically*/
11 if new time window & accuracy < threshold then
12     update_model(LR);
```

where m is the number of samples, and $x^{(i)}$ and $y^{(i)}$ represent the $\mathtt{i}$-th input sample and actual label, respectively. To minimize the negative log-likelihood loss, gradient descent is used to update the model parameters θ iteratively.

2) Interacting: After the pre-training phase, the pre-trained LR-based coupled block selection model is available. When the CoupledCB is activated, the model will be utilized to estimate the probability $P_{invalid}$ that the valid data of a block will be invalidated. To maintain the exploration ability of the proposed model, we design to randomly select coupled block based on $P_{invalid}$. That is to say, regarding a block that has a higher $P_{invalid}$, the probability of it being selected as coupled block will become lower, and vice versa.

3) Updating: The workloads generally have complicated access patterns, that is, the pre-trained model cannot be applied for all scenarios. For further improving the applicability of the LR model in CoupledCB, we design an updating phase for periodically adjusting the LR model based on the specific workload. Since the input parameters and the true labels can be collected in the runtime SSD system, the implementation of periodic updates is feasible. Note that, when a period is over, it firstly checks the accuracy of the current period. The updating process is only enabled when the accuracy is lower than the predefined threshold.

D. Implementation

For clearly illustrating the workflow of the coupled block selection in CoupledCB, Algorithm 1 shows the implementation details. First, the pre-trained LR model is introduced for checking the candidate coupled blocks, shown in line 1. Then, lines 2-9 show the specification of model checking during the GC greedy algorithm [23], [33]. The output of the LR model is the probability of the checked block, i.e., $P_{invalid}$. We use a random value for determining the block selection [31], [34], instead of setting a pre-defined threshold, as seen in lines 6-10. Note that, the selection of blocks still follows the GC greedy policy, that is, blocks with the higher number of invalid pages will ultimately be retained, as seen in lines 8-9. In addition, a pre-defined number of coupled blocks are maintained during the GC greedy algorithm. When a new time window starts, the LR model should be updated based on the access feature during the last time window, as seen in lines 10-12.

TABLE I
EXPERIMENTAL SETTINGS OF THE EVALUATED SSDS

SSD parameters	
(Channel, Chip)	`(4,2)`
(FTL, GC Threshold)	`(Page-level, 5%)`
Chip parameters	
(Die, Plane, Block, Page)	`(1, 1, 512, 512)`
(Page Size, Cell Density)	`(8KB, TLC)`
(Read, Program, Erase Latency)	`(90us, 1.1ms, 10ms)`

TABLE II
SPECIFICATIONS ON SELECTED DISK TRACES (ORDERED BY WRITE RATIO)

Trace	# of Req.	Write R	Avg. write size	Update R
$ali2$	2,434,937	84.7%	13.5KB	24.3%
$stg0$	2,030,915	84.8%	9.2KB	66.3%
$proj0$	4,224,524	87.5%	40.9KB	35.2%
$ali77$	2,724,654	93.9%	13.6KB	33.3%
$prxy0$	12,518,968	96.9%	4.6KB	55.6%
$ali181$	7,847,267	99.4%	4.9KB	56.8%

IV. EXPERIMENT AND EVALUATION

A. Experiment Setup

We conduct trace-driven simulations with *SSDsim* [13] to model 3D TLC NAND flash-based SSDs. The SSDsim simulator has been used in several SSD-related optimization studies [2], [16], [28], [33]. The evaluated SSD and chip parameters are based on [17], [20] with the detailed specifications described in Table I. We extend SSDsim to support the FSP scheme, referred to [16], [21]. Dynamic page allocation [20], greedy garbage collection [23], and static wear-leveling [35] are employed by default. To trigger enough GC operations, we set a limited capacity of 16GB SSD.

We evaluate six commonly used disk traces, and the detailed specifications of the block I/O trace are shown in Table II. Specifically, two were collected by *Microsoft Research Cambridge* [36], and four were obtained from *Alibaba Cloud* [37], representing the first *seventy-two-hour* data traces. Besides, the training datasets are also collected while replaying the other traces from *Microsoft* and *Ali*, totally 30 workloads.

The evaluation considers the following comparison counterparts to validate the effectiveness of *CoupledCB*.

- *Baseline*, which implements the default FSP scheme and data movement of copyback.
- *OSPADA* [20], an FSP optimization method that groups inconsistent access addresses into an FSP Unit to maximize the internal parallelism of SSDs.
- *PcGC* [18], which sets two pointers in the current GC victim block and then matches the odd/even sequence between the victim page and the destination page.
  PcGC is the most relevant work as it addresses the OE constraint in the GC victim block. Although it is not designed for FSP-enabled SSDs, we have implemented its principles in the context of FSP.

The time window of CoupledCB is set to 1,024, referred to [38], and the selected number of coupled blocks is 3. Additionally, the learning rate and iteration number of the LR model are set to 0.05 and 20, respectively. The threshold for activating the updating phase is 0.9.

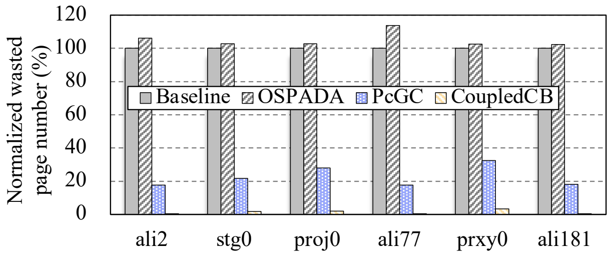

Fig. 5. The comparison of the wasted page number after running the selected block I/O traces.

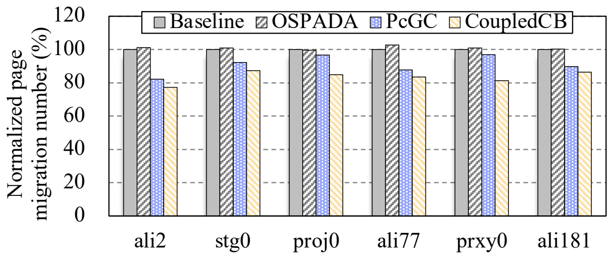

Fig. 6. The comparison of the total page migration number during the garbage collections.

B. Evaluation and Result

We use the following metrics to measure the validity of the proposed mechanism: a) *GC efficiency* and b) *I/O latency*.

1) GC Efficiency: The goal of CoupledCB is to reduce the number of wasted pages induced by copyback-based garbage collection, thus improving the GC efficiency. Figure 5 shows the statistics of wasted pages. Overall, CoupledCB outperforms Baseline, OSPADA, and PcGC by reducing the number of wasted pages by 98.6%, 98.7%, and 94.7% respectively. This is attributed to CoupledCB's ability to efficiently utilize coupled blocks to fill up wasted spaces. Only 1.4% of pages are wasted in CoupledCB, thus preventing unnecessary page migrations in advance.

Since OSPADA redistributes the data pages with consistent addresses, some valid data are left as valid pages in 3-page FSP units. Consequently, it induces the more wasted pages in contrast to Baseline, caused by worse matching in OE constraint. Additionally, PcGC can reduce the number of wasted pages to an average of 22.6% compared with the baseline method. The remaining wasted pages are due to the mismatch of valid odd and even pages in GC victim blocks and 3-page FSP granularity (as explained in Section II-A).

Improved space utilization leads to enhanced GC efficiency. As shown in Figures 6, CoupledCB outperforms the other methods, reducing page movements by 19.0%, 19.7%, and 10.8% compared to Baseline, OSPADA, and PcGC, respectively. The wasted space caused by copyback is effectively minimized by coupled blocks, indicating the data in coupled blocks is accurately predicted by the proposed LR model.

2) I/O and Tail Latency: Using coupled blocks to fill up the wasted space can improve the GC efficiency, and further mitigate delays in I/O processing. This section presents the

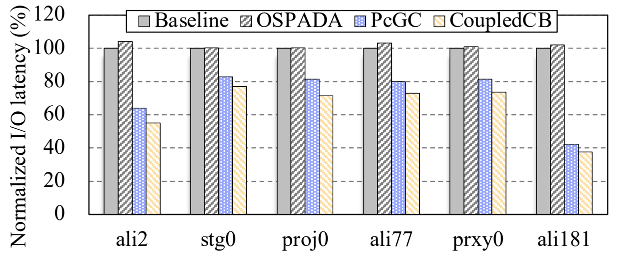

Fig. 7. The comparison of the overall I/O performance after running the selected block I/O traces.

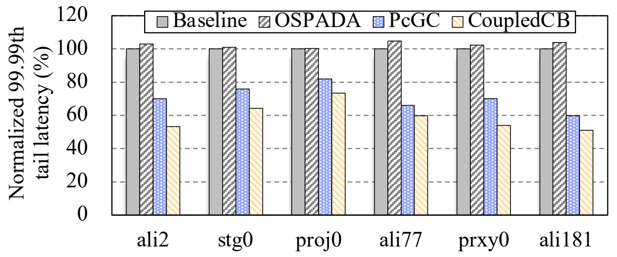

Fig. 8. The comparison of 99.99th tail latency after running the selected block I/O traces.

effect of I/O-related metrics. Figure 7 shows the I/O latency comparison. CoupledCB can greatly reduce the I/O latency by 34.9%, 35.9%, and 9.8% on average, compared with Baseline, OSPADA, and PcGC. This is because CoupledCB effectively fills the wasted space caused by the OE constraint, which in turn reduces the number of page movements via copyback. While copyback-based garbage collection delays I/O processing on the GC victim plane, eliminating page movements can directly reduce the I/O latency.

Tail latency is an important measure that focuses on the slowest I/O performance in SSDs [39], [40]. The comparison in Figure 8 illustrates the 99.99th percentile tail latency. CoupledCB can significantly reduce the tail latency by an average of 40.7%, 42.1%, and 16.2% compared to Baseline, OSPADA, and PcGC, respectively. It is evident that CoupledCB can bring about greater improvements in overall I/O latency, particularly in the context of GC operations, which can be inferred from the tail latency. This is because the slowest I/Os are mainly caused by delays in GC. Therefore, enhancing the GC efficiency can greatly reduce tail latency.

C. LR Model Accuracy

The proposed LR model predicts the probability of invalidity, denoted as $P_{invalid}$, for candidate coupled blocks. It aims to ensure that data that has been proactively moved remains unchanged until the next GC operation. Therefore, accuracy is defined as the number of unchanged moved pages divided by the total number of movements in CoupledCB. Table III presents the prediction accuracy results. The proposed LR model demonstrates an average accuracy of 97.8%. This high accuracy is attributed to its ability to effectively learn the general principles during the training phase and subsequently refine specific patterns during the updating phase.

TABLE III
THE PREDICTION ACCURACY OF $P_{invalid}$ ESTIMATED BY LR MODEL

Trace	$ali2$	$stg0$	$proj0$	$ali77$	$prxy0$	$ali181$
Acc.	98.1%	97.6%	96.9%	98.6%	97.1%	98.7%

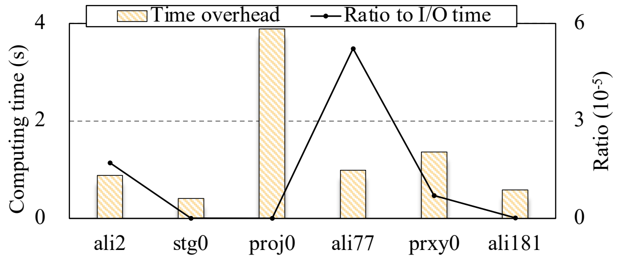

Fig. 9. The time overhead of the proposed method CoupledCB.

D. Overhead Analysis

The overheads of CoupledCB include execution time and space consumption overhead. The execution time overhead is mainly due to the implementation of the LR model. Because the model is pre-trained, the time overhead only occurs during the interacting and updating phases. Considering SSD controllers usually have limited computation power and memory capacity, the time overhead was measured on a resource-limited ARM platform equipped with an ARM Cortex-A9 Dual-Core of 666MHZ. As shown in Figure 9, CoupledCB requires less than 3.9 seconds for all traces, corresponding to 0.001% of the total I/O processing time on average.

Regarding space overhead, Section III-C1 outlines eight types of inputs, out of which four require only a single variable of 4 bytes each. Among the remaining four types, two require sets of plane numbers with 8 planes, and the other two require sets of block numbers with 4,096 elements each. Consequently, the total size of input parameters for the LR model is 32.08 kilobytes. Additionally, the weight parameters of the LR model are 36 bytes. In summary, CoupledCB requires minimal computing time and occupies very little memory space.

V. Conclusion

This paper introduces an efficient method for moving pages in copyback-based garbage collection. The method utilizes coupled blocks to make use of space that would otherwise be wasted due to the OE constraint, while also supporting preemptive data movements within the coupled blocks. To prevent inefficient proactive movements, a logistic regression-based coupled block selection model is introduced to assist in determining the coupled blocks. Evaluations show that our proposal can significantly reduce space wastage by an average of 97.4% and overall I/O latency by up to 63.1% compared to state-of-the-art schemes.

Acknowledgment

This work was partially supported by National Key R&D Program of China Grant No. 2023YFB4404400, National Natural Science Foundation of China Grant No. 62102193, State Key Laboratory of Computer Architecture (ICT CAS) Grant No. CARCH202106, Natural Science Research Start-up Foundation of Recruiting Talents of Nanjing University of Posts and Telecommunications Grant No. NY224062.

References

[1] T. Lange, J. Naor, and G. Yadgar, "Offline and online algorithms for ssd management," *Communications of the ACM*, vol. 66, no. 7, pp. 129–137, 2023.

[2] F. Wu, J. Zhou, S. Wang *et al.*, "Fastgc: Accelerate garbage collection via an efficient copyback-based data migration in ssds," in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018, pp. 1–6.

[3] J. Kim, K. Lim, Y. Jung *et al.*, "Alleviating garbage collection interference through spatial separation in all flash arrays," in *2019 USENIX Annual Technical Conference (USENIX ATC)*, 2019, pp. 799–812.

[4] R. Micheloni, "Solid-state drive (ssd): A nonvolatile storage system," *Proceedings of the IEEE*, vol. 105, no. 4, pp. 583–588, 2017.

[5] C.-Y. Liu, J. Kotra, M. Jung *et al.*, "{PEN}: Design and evaluation of {Partial-Erase} for 3d {NAND-Based} high density {SSDs}," in *16th USENIX Conference on File and Storage Technologies (FAST)*, 2018, pp. 67–82.

[6] S. Li, W. Tong, J. Liu *et al.*, "Accelerating garbage collection for 3d mlc flash memory with slc blocks," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[7] W. Kang, D. Shin, and S. Yoo, "Reinforcement learning-assisted garbage collection to mitigate long-tail latency in ssd," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–20, 2017.

[8] W. Choi, M. Jung, M. Kandemir *et al.*, "Parallelizing garbage collection with i/o to improve flash resource utilization," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2018, pp. 243–254.

[9] Micron, "Nand flash performance improvement using internal data move," http://download.micron.com/pdf/technotes/tn2915.pdf.

[10] W. Wang and T. Xie, "Pcftl: A plane-centric flash translation layer utilizing copy-back operations," *IEEE Transactions on parallel and distributed systems (TPDS)*, vol. 26, no. 12, pp. 3420–3432, 2014.

[11] TOSHIBA, "Nand memory toggle ddr1.0 technical data sheet," 2013.

[12] ONFI, "Open nand flash interface specification revision 5.0," https://www.onfi.org/specifications, 2021.

[13] Y. Hu, H. Jiang, D. Feng *et al.*, "Exploring and exploiting the multilevel parallelism inside ssds for improved performance and endurance," *IEEE Transactions on Computers (TC)*, vol. 62, no. 6, pp. 1141–1155, 2012.

[14] Toshiba, "Bics flash," https://business.kioxia.com/en-us/memory/bics.html.

[15] Hynix, "H27qfg8nnm8r bcg v3 256gb tlc nand flash toggle technical data sheet," 2016.

[16] C.-Y. Liu, Y. Lee, W. Choi *et al.*, "Gssa: A resource allocation scheme customized for 3d nand ssds," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 426–439.

[17] W. Zhang, Q. Cao, H. Jiang *et al.*, "Improving overall performance of tlc ssd by exploiting dissimilarity of flash pages," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 2, pp. 332–346, 2020.

[18] S. Pang, Y. Deng, G. Zhang *et al.*, "Pcgc: A parity-check garbage collection for boosting 3-d nand flash performance," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 4364–4377, 2023.

[19] Q. Xiong, F. Wu, Z. Lu *et al.*, "Characterizing 3d floating gate nand flash: Observations, analyses, and implications," *ACM Transactions on Storage (TOS)*, vol. 14, no. 2, pp. 1–31, 2018.

[20] F. Wu, Z. Lu, Y. Zhou *et al.*, "Ospada: One-shot programming aware data allocation policy to improve 3d nand flash read performance," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 51–58.

[21] J. Li, Z. Cai, B. Gerofi *et al.*, "Page type-aware full-sequence program scheduling via reinforcement learning in high density ssds," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 3696–3707, 2024.

[22] S. Pang, Y. Deng, G. Zhang *et al.*, "Psa-cache: A page-state-aware cache scheme for boosting 3d nand flash performance," *ACM Transactions on Storage*, vol. 19, no. 2, pp. 1–27, 2023.

[23] C. Gao, M. Ye, Q. Li *et al.*, "Constructing large, durable and fast ssd system via reprogramming 3d tlc flash memory," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, pp. 493–505.

[24] G. O. Ganfure, C.-F. Wu, Y.-H. Chang *et al.*, "Deepprefetcher: A deep learning framework for data prefetching in flash storage devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3311–3322, 2020.

[25] C. Chakraborttii and H. Litz, "Learning i/o access patterns to improve prefetching in ssds," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 427–443.

[26] W. Liu, J. Cui, T. Li *et al.*, "A space-efficient fair cache scheme based on machine learning for nvme ssds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 383–399, 2022.

[27] J. Li, B. Huang, Z. Sha *et al.*, "Mitigating negative impacts of read disturb in ssds," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 1, pp. 1–24, 2020.

[28] C. Wu, C. Ji, Q. Li *et al.*, "Maximizing i/o throughput and minimizing performance variation via reinforcement learning based i/o merging for ssds," *IEEE Transactions on Computers (TC)*, vol. 69, no. 1, pp. 72–86, 2019.

[29] Q. Wei, Y. Li, Z. Jia *et al.*, "Reinforcement learning-assisted management for convertible ssds," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[30] M. Li, C. Wu, C. Gao *et al.*, "Rlalloc: A deep reinforcement learning-assisted resource allocation framework for enhanced both i/o throughput and qos performance of multi-streamed ssds," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[31] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[32] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[33] Y. Lv, L. Shi, Q. Li *et al.*, "Mgc: Multiple-gray-code for 3d nand flash based high-density ssds," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 122–136.

[34] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[36] M. Kwon, J. Zhang, G. Park *et al.*, "Tracetracker: Hardware/software co-evaluation for large-scale i/o workload reconstruction," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2017, pp. 87–96.

[37] "Alibaba block traces," https://github.com/alibaba/block-traces.

[38] X. Xu, Z. Cai, J. Liao *et al.*, "Frequent access pattern-based prefetching inside of solid-state drives," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 720–725.

[39] S. Yan, H. Li, M. Hao *et al.*, "{Tiny-Tail} flash:{Near-Perfect} elimination of garbage collection tail latencies in {NAND}{SSDs}," in *15th USENIX Conference on File and Storage Technologies (FAST)*, 2017, pp. 15–28.

[40] W. Kang and S. Yoo, "Dynamic management of key states for reinforcement learning-assisted garbage collection to reduce long tail latency in ssd," in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018, pp. 1–6.