Separating Frozen Pages via Learning-based Recognition with ZNS SSD for Write Amplification Reduction in Database

Shikai Tan[1], Zhipeng Tan[1(✉)], Zijian Zhang[1], Wenjie Qi[1], Ying Yuan[1], and Yuan Feng[2]

[1] WNLO, Huazhong University of Science and Technology, Wuhan, China
{shikaitan, tanzhipeng, zjzhang, hustqwj, yuanying}@hust.edu.cn
[2] Wuhan Dameng Database Co., Ltd., Wuhan, China
fengyuan@dameng.com

Abstract. NAND flash-based solid-state drives (SSDs) are becoming the predominant storage medium for database systems. However, the significant mixing of pages with varying lifespans in B+-tree-based databases exacerbates the problem of huge write amplification (WA) which is caused by the rewrites of live pages during garbage collection (GC) in SSDs, reducing database performance significantly. *Frozen pages* are dominant in database workloads, which are actively updated for a period and then eventually no longer updated but remain valid. Existing data placement schemes fail to recognize and isolate frozen pages, leading to repeated migrations of them.

This paper proposes *SepFrozen*, a novel machine learning-based mechanism for recognizing and isolating frozen pages to decrease repeated migrations of them, thereby reducing WA and enhancing database performance. SepFrozen can be integrated seamlessly with existing rule-based data placement schemes. We integrate two state-of-the-art schemes with SepFrozen and implement them in LeanStore with ZNS SSDs. Experimental results show that SepFrozen reduces WA by 30.2% and enhances database throughput by 19.5%.

Keywords: Write Amplification · ZNS SSD · Frozen Page · Machine Learning

1 Introduction

NAND flash-based solid state drives (SSDs) are becoming the predominant persistent storage medium for database systems due to their increased performance and reduced cost [1]. However, Write Amplification (WA) [3] arises when SSDs are used as storage devices for databases [2,9], which means the actual amount of data written to the underlying flash medium exceeds the amount of data written by the database [2]. High WA degrades SSD lifetime and I/O performance [12] due to excessive writes, thus reducing database performance [2]. WA is caused by additional Garbage Collection (GC) operations in SSDs due to the "out-of-place update" [6] characteristic. In addition, the significant mixing of database

pages with varying lifespans further exacerbates this phenomenon [2], because of rewrites of cold valid pages when collecting hot obsolete pages. In this paper, we define the **lifespan** of a page (4 KiB) as the number of pages written by the database from when a page is written until it is invalidated (or until the end of the benchmark). We employ the term **hotness** to quantify the duration of a page's lifespan: hot pages are characterized by shorter lifespans, and vice versa.

A key technique to reduce WA is the isolated placement of data based on their invalidation times [3]. Specifically, pages with similar invalidation time are written into the same erase block, allowing them to expire around the same time, reducing the proportion of valid pages in the block for GC, thereby lowering WA. However, existing solutions [3–8] are primarily designed for general block I/O workloads and lack adaptation to the characteristics of databases. For instance, the issue of **frozen pages** [2,10,11] exists under database workloads, where many pages are frequently rewritten to the SSD within a short time window, and do not get updated afterward but remain valid. There are two significant shortcomings in current research leading to unnecessary migrations of frozen pages. First, they do not implement efficient frozen page isolation mechanisms [3,4,7]. They depend on page hotness for placement, and once a page transitions to a frozen state, its hotness gradually diminishes, instead of being directly allocated to a designated area for frozen pages. Second, existing studies [2, 10] do not fully leverage page characteristics and thus lack effective methods for recognizing frozen pages.

In this paper, we propose a learning-based frozen page recognition and isolation mechanism (**SepFrozen**) aiming at reducing WA for databases. The key insight of SepFrozen is to utilize an effective frozen page recognition model to extract frozen pages from valid pages during GC and place them in an isolated area, avoiding mixing with normal pages. SepFrozen can be well integrated with existing rule-based data placement schemes. We integrate two state-of-the-art rule-based data placement schemes (SepBIT [3] and DAC [4]) with SepFrozen, resulting in two frozen-page aware data placement schemes (SepFrozenBIT and SepFrozenDAC) that demonstrate enhanced performance.

Our schemes necessitate the recognition of frozen pages based on their characteristics on the host side, as well as the control over data placement and GC. However, conventional block interface SSDs, through their Flash Translation Layer (FTL), conceal the actual physical addresses of pages [12], thereby failing to fulfill this requirement. Emerging Zoned Namespace (ZNS) SSDs [12] enable fine-grained data placement and GC on the host side. Consequently, this paper implements frozen page-aware data placement schemes in LeanStore [1] (a B+-tree-based database storage engine) with end-to-end support for ZNS SSDs, to achieve the goal of reducing WA in databases.

The main contributions of the paper are summarized as follows:

1. We propose a lightweight and effective machine learning-based frozen page recognition model which can extract frozen pages from valid pages accurately.

2. We propose a novel frozen page isolation mechanism (SepFrozen) which can be integrated with various existing rule-based data placement schemes seamlessly.

3. We integrate two state-of-the-art schemes with SepFrozen and thus propose two frozen page-aware data placement schemes, which proactively recognize and isolate frozen pages. We implement them in LeanStore with end-to-end support for ZNS SSDs.

4. We evaluate SepFrozen using TPC-C and YCSB workloads, demonstrating that SepFrozen effectively reduces the write amplification and improves the performance of databases.

This paper is organized as follows. Section 2 introduces the background and related work. Section 3 describes the motivation of our work. Section 4 details the design and implementation of our schemes. Section 5 evaluates the performance of our schemes, and we conclude this paper in Section 6.

2 Background and Related Work

2.1 Background

Writing to flash pages (flash read/write units) necessitates first resetting the chip at a larger granularity of "erase block" (a.k.a. superblock) before new data can be written [6]. Due to the "erase before write" restriction, SSDs perform garbage collection (GC), copying valid pages in the flash block to a new location before erasing and writing new data [2]. GC results in additional writes to flash pages, leading to WA in SSDs [12]. In databases based on B+-tree index, the characteristics of "write skew" and "temporal locality" result in severe mixing of hot and cold data [2], further intensifying the WA problem. We define the **Write Amplification Factor (WAF)** [3] as the ratio between the actual amount of data written to the underlying flash storage and the data written by the host (i.e., the database). Studies [2, 9] show that during TPC-C benchmarking on SSDs, WAF can increase to over 4 times its initial value, resulting in declining SSD IOPS and a reduction in tpmC to approximately 1/3 of its original level.

LeanStore [1] is a high-performance B+-tree-based database storage engine designed for modern CPUs and NVMe SSDs.

2.2 Related Work

Data Placement Schemes. To perform data separation, we need to predict when a page will be overwritten in the future. Most prior works in this respect are rule-based [2–5, 9] and learning-based [6–8] schemes. Rule-based schemes are simpler to implement and have lower overhead. SepBIT [3] and DAC [4] separate data based on inferred lifespan and write count, respectively. 2R [2] proposes "two region" FTL and isolate cold pages into the cold flash region. WARCIP [5] minimizes the rewrite interval variance of pages in a flash block through Greedy Clustering. FlashAlloc [9] de-multiplexes concurrent writes into per-object dedicated flash blocks. However, since the database file is a single large object, this approach is difficult to work effectively. Learning-based schemes offer better adaptability and more accurate hot/cold separation, albeit with higher overhead [6]. PHFTL [6] and ML-DT [8] employ GRU and TCN models to predict

page lifetimes, respectively. MiDAS [7] employs MCAM to dynamically adjust the number of groups and sizes. However, both rule-based and learning-based schemes fail to recognize and isolate frozen pages, leading to repeated migrations of them. Therefore, we develop a learning-based frozen page recognition model built upon rule-based schemes, leveraging the low overhead of rule-based schemes and the high accuracy of learning-based approaches.

ZNS SSD. Conventional block interface SSDs, through their FTL, conceal the actual physical addresses of pages. As a result, existing research employs SSDs with novel interfaces to enable host-side control over data placement, such as Multi-stream SSDs [10], Open-Channel SSDs [14], and OpenSSDs [6]. Nevertheless, Multi-stream SSDs cannot fully coordinate GC between the host and the SSD, while Open-Channel SSDs and OpenSSDs lack a unified standard in the industry, leading to high adaptation and maintenance costs. The NVMe Zoned Namespace (ZNS) [12] is a new storage interface that groups logical blocks into zones and mandates sequential writes for each zone (while still allowing random reads). ZNS SSDs enable fine-grained data placement and GC on the host side [12]. This enables the host to optimize the writing schemes based on workload characteristics more efficiently, further reducing GC costs and WA. Additionally, there are commercial ZNS SSD products [13] on the market. Therefore, this paper integrates our proposed frozen page-aware data placement schemes with ZNS SSDs to reduce WA in databases.

3 Motivation

In this section, we first demonstrate the mixing of hot/cold data of the database, and then identify the limitations of existing studies.

We perform the TPC-C benchmark on LeanStore and then calculate the lifespan for each page with the unit of bytes. Fig. 1 shows the page lifespan distributions over logical block addresses (LBAs), with the background color indicating the frequency of the corresponding lifespan values. The page hotness exhibits a two-level distribution, where hot pages with shorter lifespans and cold pages with longer lifespans are mixed, significantly increasing WA during GC.

We evaluate the WA optimization effects of four rule-based data placement schemes (SepBIT, DAC, 2R, WARCIP) and NoSep scheme (no data separation) in database workloads, as depicted in Fig. 3. To highlight the performance shortcomings of current schemes, we employ the **FK** scheme [3] which represents an oracular baseline that leverages future knowledge for placement decisions: running the benchmark once in advance, calculating the lifespan of each page, and then using K-means clustering to group pages based on their lifespans [7]. We found that, even the top-performing scheme, SepBIT, exhibits a 43.3% gap in WA compared to FK, indicating that current data placement schemes still have substantial room for improvement in database scenarios. Next, we will analyze the two causes of this performance gap.

Cause 1: data placement schemes lack awareness of database frozen pages. Frozen pages play a crucial role in database workloads; however, exist-

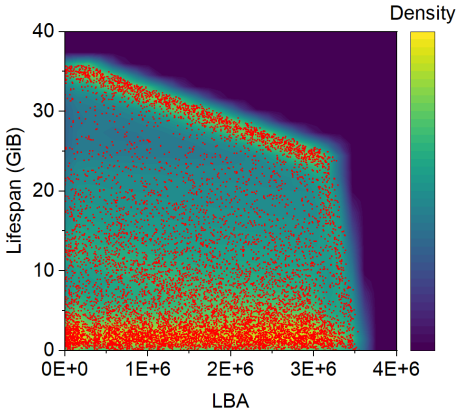

Fig. 1: Lifespan distribution of TPC-C workload over logical block addresses.

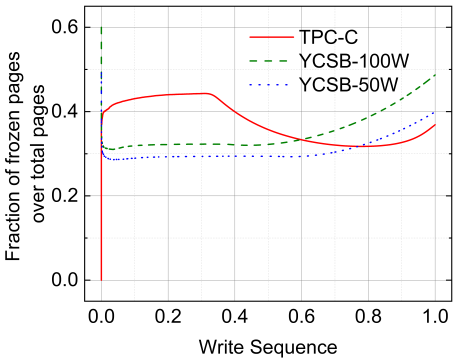

Fig. 2: Frozen pages in database workloads.

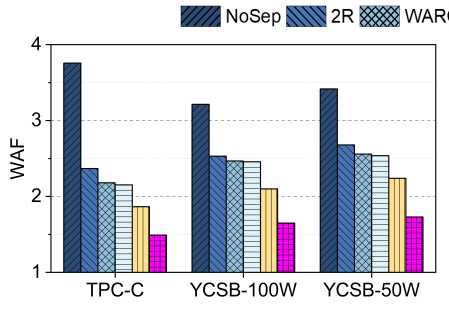

Fig. 3: Overall WAF of existing schemes.

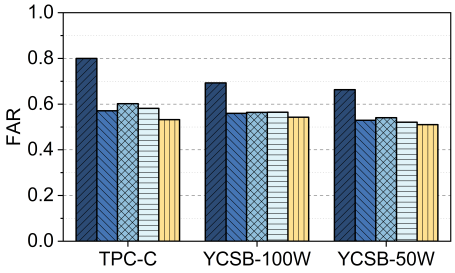

Fig. 4: Overall FAR of existing schemes.

ing data placement schemes do not account for them, resulting in unnecessary repeated migrations.

Frozen pages are dominant in database workloads. We observed that many pages are frequently updated and written to the SSD within a short time window, but then eventually are no longer updated and remain valid. In relevant research [2, 10], these pages are referred to as **frozen pages**. Fig. 2 illustrates the fraction of frozen pages over time. Both the x-axis and y-axis have been normalized to a range of 0 to 1, clearly demonstrating that frozen pages constitute around 30% to 50% of the cumulative written pages.

Data placement schemes have repeated migrations for frozen pages. Existing data placement schemes depend on page hotness for placement. Once a page transitions to a frozen state, its hotness gradually diminishes, instead of being directly allocated to a designated area for frozen pages. We now explain it by analyzing SepBIT and DAC.

SepBIT: Victim pages of varying ages (the total number of pages written to the SSD since last write) are migrated to different zones. As illustrated in Fig. 5, a possible scenario is that a page is initially written to the zone of class 1 (the hottest), then becomes a frozen page, after which it is moved to the zone of class 3 during GC. As the page ages, it may be migrated to the zone of class 4 and 5 subsequently, ultimately reaching the coldest zone (class 6).

DAC: The page is migrated to a hotter zone with each database write and a colder zone during GC, as depicted in Fig. 6. A possible scenario is that a page may be written to the zone of class 6 (the hottest) before becoming a frozen page, after which it is moved to the zone of class 5, 4, 3, 2 sequentially before finally reaching class 1 (the coldest) each time it is selected as a victim page.

We define the Frozen Page Amplification Rate (**FAR**) as the proportion of frozen pages among all valid pages migrated during GC. Fig. 4 shows the FAR of the four rule-based schemes (SepBIT, DAC, 2R, WARCIP) and NoSep. All of them exhibit a FAR of above 50%, indicating that for every 100 victim pages, at least 50 frozen pages are migrated repeatedly.

Cause 2: lack of effective frozen page recognition methods. The premise of proactively migrating frozen pages in isolation is to effectively recognize frozen pages. The difficulty lies in how to fully leverage page characteristics for accurate recognition. Currently, only 2R [2] has proposed a method for recognizing frozen pages, which simply classifies pages written by the database as normal pages and pages written during GC as frozen pages. This method fails to fully leverage the hotness characteristics of pages, resulting in substantial misclassification of normal pages as frozen pages (as shown in Fig. 18, the false positive rate(FPR) is 41%). Another study [10] employs a hard-coded method to recognize frozen pages based on attributes such as the tables to which the pages belong. However, this is not a general method for different scenarios.

In light of the above analyses, there is an urgent need for an effective frozen page recognition method and a frozen page-aware data placement scheme. We will now provide a detailed description of our proposed solutions.

4 Design and Implementation

In this section, we provide detailed descriptions of SepFrozen, including the learning-based frozen page recognition model and the frozen page-aware data placement schemes. Finally, we deploy our schemes in LeanStore with ZNS SSD.

In order to accurately and efficiently recognize frozen pages, we propose a learning-based frozen page recognition model. We first extract six features that represent the key attributes of frozen pages, then collect and process the training data. Subsequently, we train a logistic regression model with the Stochastic Gradient Descent algorithm and deploy it in the system.

To effectively mitigate the issue of repeated migrations of frozen pages in existing data placement schemes, we propose a novel frozen page isolation mechanism (SepFrozen) which can be integrated with various existing rule-based data placement schemes seamlessly. We integrate two state-of-the-art schemes with

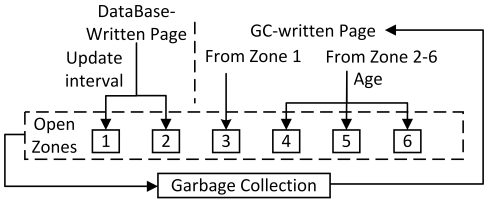

Fig. 5: SepBIT workflow.

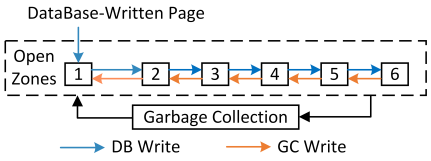

Fig. 6: DAC workflow.

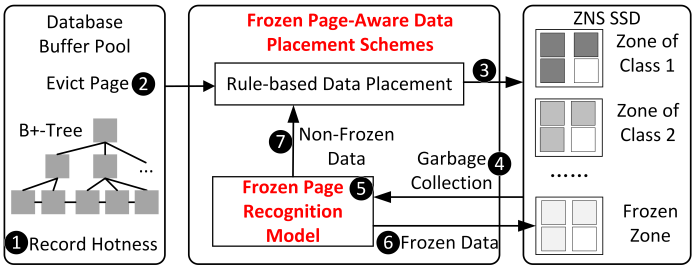

Fig. 7: SepFrozen Overview.

SepFrozen and thus propose two frozen page-aware data placement schemes, which proactively recognize and isolate frozen pages. The workflow of SepFrozen is shown in Fig. 7 and includes the following steps: ① During database operation, page hotness information is recorded within the page. ② Evicted pages are written to ZNS SSD. ③ Based on the rule-based data placement scheme, pages are assigned to different zones. ④ When storage space is insufficient, GC is performed and valid pages are read into memory. ⑤ Hotness records are extracted from the pages, and the frozen page recognition model is used to determine whether a page is frozen. ⑥ If the page is determined to be frozen, it is written directly to the frozen zone; ⑦ otherwise, it is placed in the appropriate zone according to the rule-based placement scheme.

4.1 Learning-based Frozen Page Recognition Model

The process comprises four stages: feature extraction, data processing, model training, and online inference.

Feature Extraction. We utilize a monotonic *timer* (instead of the real timestamp) that increments by one for each database-written page. For each page, we extract six features: write time (WT), valid data in the page (VD), average access interval (AI), average modify interval (MI), access count (AC), and modify count (MC). We collectively refer to the tuple of (WT, VD, AI, MI, AC, MC) as a **hotness record** (HR). WT denotes the time at which a page is written to the SSD. The update interval between two consecutive writes effectively characterizes the hotness of the page. VD indicates the total number

of bytes of valid records in the page, which remains unchanged once the page becomes frozen. We track the access and modification times for each page in memory, from the last read into memory until its subsequent write to the SSD. AC and MC represent the access and modification counts, respectively. AI and MI are the averages of the access and modification intervals, respectively. AI, MI, AC and MC provide a robust indication of the hotness of pages in memory. We calculate the Pearson correlation coefficients between the frozen attribute and the following six attributes: the difference between the two consecutive WT values of the same page (WT-diff), the difference between the two consecutive VD values of the same page (VD-diff), AI, MI, AC, and MC. The resulting coefficients are 0.7369, 0.7948, 0.5021, 0.4937, 0.6374 and 0.5695, respectively. All correlation coefficients are higher than 0.4, indicating that the six variables are correlated with the frozen attribute to a certain extent.

Data Processing. This stage includes collecting hotness records, tagging frozen labels, standardization, packaging data, and dividing the datasets into training and testing sets. (1) During benchmarking, we collect hotness records of every database-written page, along with its invalidation time. If the page is generated by a new write, we assume its invalidation time is infinite. (2) If the invalidation time is finite, the page is not yet frozen. Otherwise, if the invalidation time is infinite, the page is tagged with a frozen label. (3) The values are standardized to enhance model performance. (4) We package the current and last hotness records of the page, referred to as $HR_{current}$ and HR_{last}, into a single sample. If the page is generated by a new write, we set all the values of HR_{last} to zero. (5) We perform random sampling, dividing the dataset into training and testing sets at a ratio of 3:1, allocating 75% of the data to the training dataset.

Model Training. We predict whether a page is frozen based on its hotness information, framing this as a binary classification problem that can be addressed using a logistic regression model [6]. The logistic regression model calculates the dot product of the hotness record and the weight vector, followed by applying the sigmoid function to transform this into a probability value between 0 and 1. If the resulting probability exceeds a predefined threshold (commonly set at 0.5), the model classifies the page as *frozen*; otherwise, the model classifies the page as *normal*. The logistic regression model employs Maximum Likelihood Estimation (MLE) to estimate the weight parameters. To further improve the model's fitting performance, we apply the Stochastic Gradient Descent (SGD) algorithm for iterative optimization of these parameters. For ease of reference, we will denote this as the **SGD** model in this paper. Given that logistic regression does not accommodate temporal feature inputs, we stack $HR_{current}$ and HR_{last} to create a single sample with 12 features. Finally, we train the SGD model using the training set. In Sec. 5.2, we compare the classification performance of the SGD model with other commonly used binary classification models to demonstrate why it is chosen for frozen page identification.

Online Inference. We implement model inference in C++ within LeanStore. The model has two types of misclassifications (Sec. 5.1): false negative (where a

frozen page is mistakenly classified as normal) and false positive (where a normal page is mistakenly classified as frozen). False negative pages will undergo multiple migrations according to SepBIT and DAC schemes, eventually reaching the coldest zone. Conversely, for pages misclassified as frozen, once the page is modified and rewritten, the original data is marked invalid, and the new page is written to a normal zone.

4.2 Frozen-page Aware Data Placement Schemes

We integrate two effective rule-based data placement schemes (SepBIT and DAC) with SepFrozen, resulting in two frozen-page aware data placement schemes (SepFrozenBIT and SepFrozenDAC) that demonstrate enhanced performance. The following outlines the specific processes of these two schemes, using six active zones as an example.

The flow of the **SepFrozenBIT** scheme is illustrated as Fig. 8. A global lifespan threshold, termed $lifespan\_threshold$, is dynamically maintained based on the average lifespans of recently reclaimed zones. The lifespan of a zone is defined as the time difference between when the zone is reclaimed and when the first page is appended to it. For database-written pages, we calculate their update intervals (the time difference between the current write and the last write). If it is issued from a new write, we assume that it has an infinite update interval. If the page's update interval is less than $lifespan\_threshold$, it is written to the zone of class 1; otherwise, it is written to the zone of class 2. For GC-written pages, we apply the frozen page recognition model to determine whether the page is frozen. If classified as frozen, it is directly written to the frozen zone (zone of class 6); otherwise, we check whether the page is from the first or second class. If from class 1, it is migrated to class 3; if from class 2, it is directed to the fourth, fifth, or sixth classes based on its age (the time difference between the current time and the time of its last write), compared against the threshold values of $4 * lifespan\_threshold$ and $16 * lifespan\_threshold$.

The flow of the **SepFrozenDAC** scheme is illustrated as Fig. 9. A temperature value is maintained for each page, starting at 2 for pages written for the first time. The temperature value increases by one with each write by the database and decreases by one with each GC. For database-written pages, if the temperature value is i, the page is written to the i-th class, and its temperature is incremented. For GC-written pages, we first use the frozen page recognition model to determine whether the page is frozen. If deemed frozen, it is written directly to the frozen zone (zone of class 1); otherwise, if its temperature value is i, it is written to the i-th zone, and its temperature is decremented by one.

4.3 Implementation Details

This paper implements frozen page-aware data placement schemes (SepFrozenBIT and SepFrozenDAC) in C++ within LeanStore (a B+-tree-based database storage engine), and establishes end-to-end integration with ZNS SSD. At first, we need to consider how to manage data read/write for ZNS SSD. Flash-Friendly

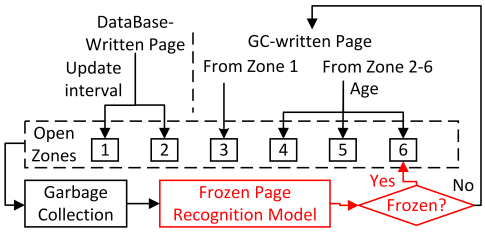

Fig. 8: SepFrozenBIT workflow.

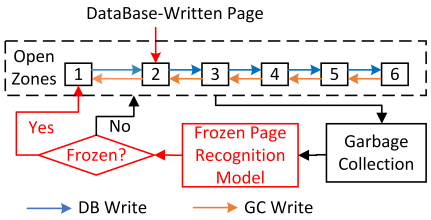

Fig. 9: SepFrozenDAC workflow.

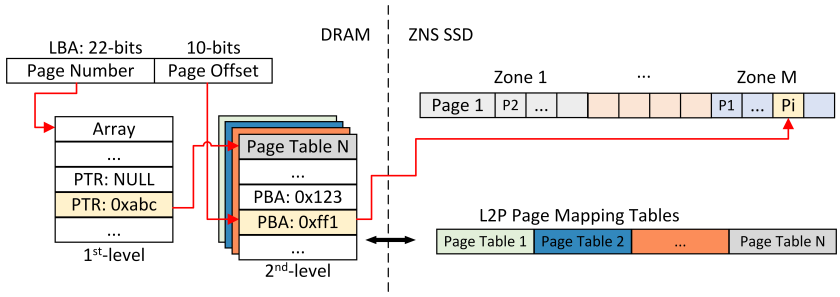

Fig. 10: Two-level L2P Page Mapping Table Hierarchy.

File System (F2FS) [15] has been adapted to support ZNS SSD devices. However, utilizing F2FS would necessitate modifications to its internal data placement and GC strategies, as well as the transmission of the database's page hotness information to F2FS, complicating the implementation. ZoneFS [16], which is specifically designed for zoned devices (such as ZNS SSDs and SMR HDDs [12]), exposes each zone as a file visible to user space. Through these zone files, applications can access each zone of the ZNS SSD using POSIX system calls (e.g., open, read, and write) [16], allowing applications to perform user-controlled data placement and GC while complying with sequential write constraints. Consequently, we implement our solution on ZoneFS.

Two-level Page Mapping. This study uses ZoneFS to manage reads and writes to ZNS SSDs and must adhere to sequential write constraints. However, B+-tree databases exhibit a highly random write pattern, necessitating the implementation of a logical-to-physical page mapping table within the database, as depicted in Fig. 10. The tables maintain the mapping from logical block addresses (LBAs) to physical block addresses (PBAs), both of which are 32 bits (capable of representing up to 16 TiB of data). We employ an array for the first-level mapping table, using the most significant 22 bits of the LBA as offsets, and the corresponding elements serving as memory addresses of the second-level mapping tables. The first-level mapping table resides in memory for quick access. Each second-level mapping table is 4 KiB in size, using the least significant 10 bits of the LBA as offsets to store corresponding PBA values. We persistently

store the second-level page tables in the order of their logical addresses, while caching accessed second-level mapping tables in memory. After getting the corresponding PBA for a given LBA, we can compute the zone number and the offset within the zone based on the PBA.

Storage Usage. We must allocate a small amount of space within each 4KiB B+ tree page to store $HR_{current}$ and HR_{last}. Each record occupies a total of 16 bytes, resulting in an overall cost of 32 bytes per page. Additionally, extra storage space is required for the secondary mapping table. For a total data volume of 1 TiB, the mapping table will require 1 GiB of storage space.

Memory Usage. For a total data volume of 1 TiB, the first-level mapping table requires 2 MiB of memory. The memory usage of the second-level mapping tables is also proportional to the dataset size. For example, in the TPC-C experiment in Sec. 5, the second-level mapping tables collectively occupy 28.16 MiB of memory. Our employed SGD model is linear and incorporates 12 feature inputs, leading to 12 corresponding weight values. Each weight, represented as *double* in C++, totals 96 bytes. During GC, batch inference is conducted, and the hotness records of all victim pages are stored in an array. For instance, to reclaim a zone of 256 MiB, assuming all pages are valid (a scenario that is not realistically achievable), the memory required for model input data would be 6144 KiB. Additionally, there is overhead associated with intermediate computational results. Empirical measurements indicate that a single inference of the model consumes 14,720 KiB of memory.

Time Overhead. Since GC is managed by a separate thread and independent of the working thread, and the system does not continuously require GC, we perform frozen page recognition on the victim pages during GC. This design not only circumvents the critical writing path but also allows some hotter pages in the victim zone to become invalid preemptively due to rewriting, thus reducing both inference overhead and the likelihood of false positives. Fig. 19 shows the inference overhead of the model. Experimental results confirm that if the model completes inference within 0.05 seconds during GC, the database performance will not be adversely affected.

5 Evaluation

5.1 Experimental Setup

Competitors. We compare SepFrozenBIT and SepFrozenDAC with SepBIT [3], DAC [4], WARCIP [5], and 2R [2], along with NoSep. NoSep appends any written pages (either database-written or GC-rewritten pages) to the same open zone. 2R classifies database-written pages as normal pages and GC-rewritten pages as frozen pages, separating normal and frozen pages into two different open zones. The number of open zones of SepBIT, DAC, and WARCIP is all set to 6.

Hardware. The experiments were performed on a Linux 5.15 system with an Intel Xeon E5-2620 v4 CPU (2.10GHz, 16 cores, 32 hardware threads), 128 GiB of main memory and a Western Digital Ultrastar DC 4TiB ZN540 ZNS SSD as storage.

Workloads. The YCSB experiment performs accesses to 10 GiB datasets and thus 336 million key-value records. We performed it with 100% writes (100W) and 50% writes mixed with 50% reads (50W). During the 60-minute test, a total of 511 GiB and 365 GiB of data were written for YCSB-100W and YCSB-50W workloads, respectively. TPC-C provides a more intricate benchmark for online transaction processing, featuring various transaction types and a sophisticated database structure. The size of the dataset is determined by the number of warehouses and expands as the benchmark runs. The number of warehouses is set to 100. As a result, a total of 406 GiB of data was written during the 120-minute test. The size of the database buffer pool is set to 1 GiB for both TPC-C and YCSB workloads.

Scheme Evaluation Metrics. We employ three metrics to assess the effectiveness of the proposed data placement schemes: FAR (Frozen Page Amplification Rate, refer to Section 3), WAF (Write Amplification Factor, refer to Section 2.1), and tpmC (transactions per minute for the TPC-C workload) as well as ops/s (operations per second for the YCSB workloads).

Model Evaluation Metrics. We use accuracy, recall, and false positive rate (FPR) to evaluate the classification performance of the frozen pages recognition models. We first label frozen pages as "positive" and normal pages as "negative". Then, we define true positives (TPs) and false negatives (FNs) as the number of frozen pages that are correctly and incorrectly classified, respectively. Similarly, we define true negatives (TNs) and false positives (FPs) as the number of normal pages that are correctly and incorrectly classified, respectively. **Accuracy**, **recall**, and **FPR** are then defined as $(TP + TN)/(TP + TN + FP + FN)$, $TP/(TP + FN)$, and $FP/(FP + TN)$, respectively.

5.2 Results

In this section, we present evaluation results. First, we compare FAR, WAF, and database throughput performance under three different database workloads across our two proposed schemes and five existing schemes. Next, we adjust various parameters in our data placement schemes and report the WAF under the TPC-C workload, illustrating the effectiveness of our schemes under different parameter configurations. Lastly, we examine the performance of frozen page recognition models and outline the determination of key settings of the model.

Default Configuration. We use Cost-Benefit [3] as our default GC policy for zone selection and fix the zone size and the GP threshold as 256 MiB and 15%, respectively. We define the garbage proportion **(GP)** as the fraction of invalid pages among all pages. When the overall GP value in the ZNS SSD is higher than the GP threshold, GC is triggered.

Performance Comparison. Fig. 11-14 illustrates the performance comparison results. Initially, SepFrozenDAC and SepFrozenBIT show a reduction in FAR of 27.5% to 35.4% relative to DAC and SepBIT, respectively, indicating that proactive frozen page recognition and isolation effectively reduce repeated page migrations. With the reduction in FAR, SepFrozenDAC and SepFrozenBIT also achieve a reduction in WAF compared to DAC and SepBIT. Notably,

the WAF of SepFrozenBIT decreases to 1.60, representing a 30.2% reduction compared to SepBIT under the TPC-C workload; in comparison, the WAF of FK that leverages future knowledge for placement decisions (see Section 3) is 1.49, whereas SepFrozenBIT's WAF is nearly identical. Finally, the effective reduction in WAF leads to increased throughput (tpmC) in SepFrozenDAC and SepFrozenBIT by 15.6% and 19.5% relative to DAC and SepBIT, respectively.

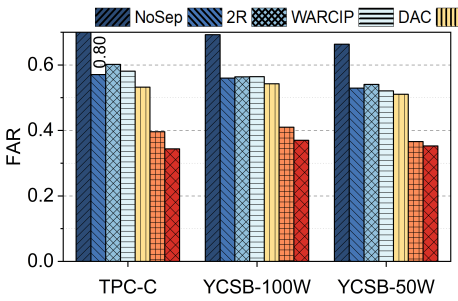

Fig. 11: Overall FAR of each scheme.

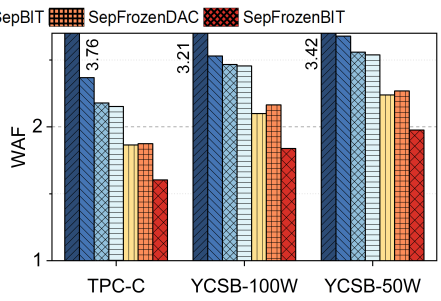

Fig. 12: Overall WAF of each scheme.

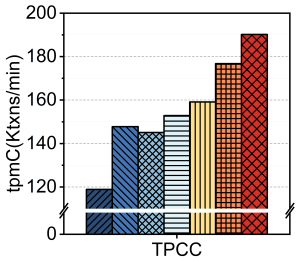

Fig. 13: Throughput under TPC-C workload.

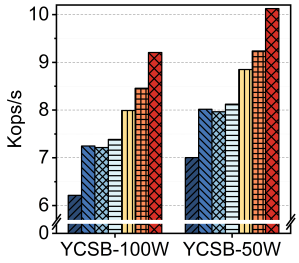

Fig. 14: Throughput under YCSB workloads.

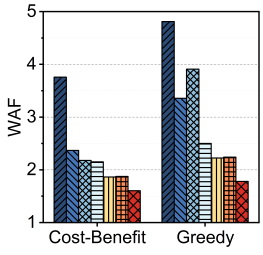

Fig. 15: WAF impact of zone selection algorithm.

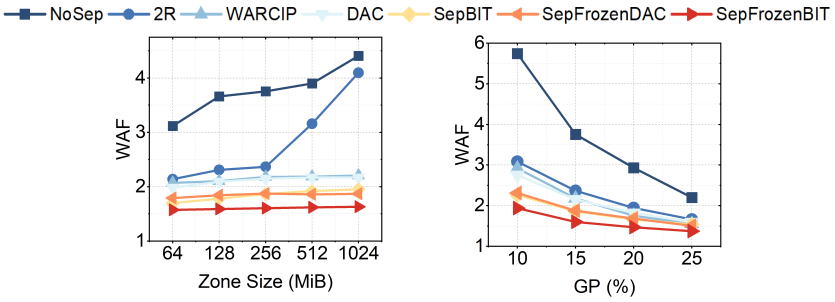

Fig. 16: WAF impact of zone sizes.

Fig. 17: WAF impact of GP thresholds.

Impact of Data Placement Scheme Configurations. Zone selection algorithm, zone size, and GP threshold significantly affect WA [3]. Therefore, in order to verify that our schemes are effective under different parameter configurations, we vary the zone selection algorithm, zone size, and GP threshold for evaluation. As indicated in Section 5.2, reduced WAF directly correlates with improved database performance. Given space limitations, we focus exclusively on the WAF results of the seven schemes under the TPC-C workload.

Configuration 1 (Zone Selection): We use Greedy [3] and Cost-Benefit [3] strategies for zone selection in GC. As illustrated in Fig. 15, Cost-Benefit yields a lower WAF than Greedy, and with both selection methods, SepFrozenDAC and SepFrozenBIT reduce WAF compared to DAC and SepBIT.

Configuration 2 (Zone Size): We vary the zone size from 64 MiB to 1024 MiB. Different zone sizes are simulated by restricting the available size of zone files. For example, when the zone size is set to 128 MiB, only the first 128 MiB of the file is accessed. As depicted in Fig. 16, WAF increases with larger zone sizes across all schemes. Notably, SepFrozenDAC and SepFrozenBIT consistently exhibit lower WAF than the other schemes, with reductions of 17.5% to 33.9% compared to DAC and SepBIT.

Configuration 3 (GP threshold): We vary the GP threshold from 10% to 25%. As demonstrated in Fig. 17, elevating the GP threshold induces a progressive decline in WAF across all schemes. Particularly, SepFrozenDAC and SepFrozenBIT consistently outperform the others, achieving WAF reductions of 14.5% to 30.8% compared to DAC and SepBIT.

Frozen Page Recognition Model Comparison. We compare the classification performance, training time and inferring time (normalized for 256 MiB of data) of the SGD model with three other commonly used binary classification models: logistic regression (LR), support vector classifier (SVC), and GRU recurrent neural network [6]. Fig. 18 and Fig. 19 shows that, while the classification performance of the SGD model is somewhat lower than that of the GRU model, its inference overhead is considerably lower. Therefore, we select the SGD model as the frozen page recognition model for its efficiency.

Key Parameter and Features Determination. We now evaluate the impact of a key parameter, the time window length (TWL), on the model's inference performance. TWL refers to the number of historical hotness records per sample of the model's input. Fig. 20 shows the model's classification performance as TWL varies from 1 to 4. Increasing TWL from 1 to 2 led to a significant improvement in classification accuracy, increasing from 78% to 92%. However, as TWL increased, storage overhead also grew linearly (see Section 4.3). Consequently, we set TWL to 2 in this study to balance model performance gains with storage efficiency. Additionally, we adjust the input feature set of the model. Initially, only the VD feature is included, followed by the sequential addition of WT, AI, MI, AC, and MC. As shown in Fig. 21, when only the VD feature is used, the model achieves an accuracy of only 71%. However, when both VD and WT are included, the accuracy increases to 89%. With all six features incorporated, the model accuracy reaches 92%.

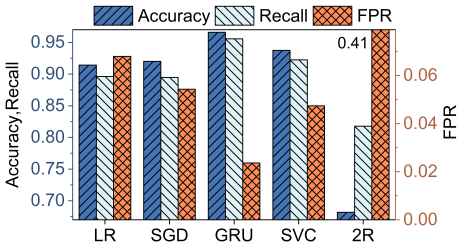

Fig. 18: Performance of Different Frozen Page Recognition Models.

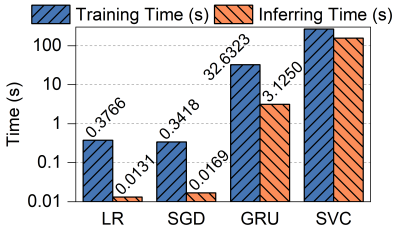

Fig. 19: Time Overhead of Different Frozen Page Recognition Models.

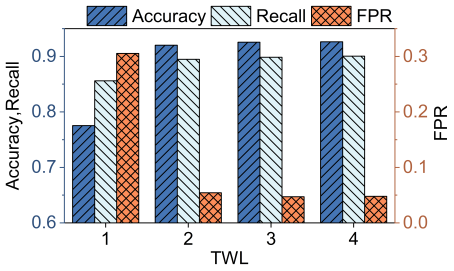

Fig. 20: Performance impact of TWL.

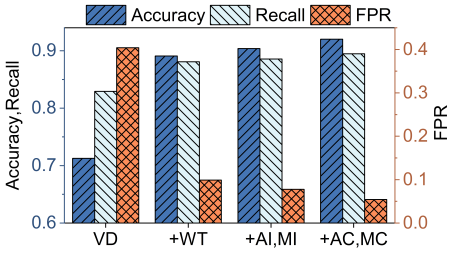

Fig. 21: Model Performance Impact of Feature Sets.

6 Conclusions

We propose SepFrozen, a novel learning-based data placement mechanism for WA reduction in databases by recognizing and isolating frozen pages. Inspired by the observation that frozen pages are dominant in database workloads, we utilize a lightweight and effective machine learning model to recognize frozen pages based on their hotness information and place them in isolation. This mechanism mitigates the issue of repeated migrations, thereby reducing WA. SepFrozen integrates seamlessly with existing rule-based data placement schemes. For database-written pages, we utilize a rule-based data placement scheme; for GC-written pages, we apply the frozen page recognition model to determine if the page is frozen. If classified as frozen, it is directly written to the frozen zone; otherwise, we continue to apply the rule-based data placement scheme. Evaluations demonstrate that SepFrozen achieves lower WA and significantly enhances database performance compared to existing data placement schemes.

Acknowledgments. We thank the anonymous reviewers for their valuable comments. This work was supported in part by research program No. 2023BAA018.

References

1. Vöhringer, D., Leis, V.: Write-Aware Timestamp Tracking: Effective and Efficient Page Replacement for Modern Hardware. Proceedings of the VLDB Endowment

16(11), 3323-3334 (2023)
2. Kang, M., Choi, S., Oh, G., Lee, S. W.. 2r: Efficiently isolating cold pages in flash storages. Proceedings of the VLDB Endowment 13(12), 2004-2017 (2020)
3. Wang, Q., Li, J., Lee, P. P., Ouyang, T., Shi, C., Huang, L.: Separating data via block invalidation time inference for write amplification reduction in Log-Structured storage. In: 2022 USENIX Conference on File and Storage Technologies (FAST 22), pp. 429-444 (2022)
4. Chiang, M. L., Lee, P. C., Chang, R. C.: Using data clustering to improve cleaning performance for flash memory. Software: Practice and Experience 29(3), 267-290 (1999)
5. Yang, J., Pei, S., Yang, Q. WARCIP: Write amplification reduction by clustering I/O pages. In: Proceedings of the 12th ACM International Conference on Systems and Storage, pp. 155-166 (2019)
6. Sun, P., You, L., Zheng, S., Zhang, W., Ma, R., Yang, J., Huang, L: Learning-based Data Separation for Write Amplification Reduction in Solid State Drives. In: 2023 60th ACM/IEEE Design Automation Conference (DAC), pp. 1-6 (2023)
7. Oh, S., Kim, J., Han, S., Kim, J., Lee, S., Noh, S. H: MIDAS: Minimizing Write Amplification in Log-Structured Systems through Adaptive Group Number and Size Configuration. In: 22nd USENIX Conference on File and Storage Technologies (FAST 24), pp. 259-275 (2024)
8. Chakraborttii, C., Litz, H: Reducing write amplification in flash by death-time prediction of logical block addresses. In: Proceedings of the 14th ACM International Conference on Systems and Storage, pp. 1-12 (2021)
9. Park, J., Choi, S., Oh, G., Im, S., Oh, M. W., Lee, S. W: FlashAlloc: Dedicating Flash Blocks by Objects. Proceedings of the VLDB Endowment 16(11), 3266-3278 (2023)
10. Park, H. W., Choi, S., An, M., Lee, S. W: Freezing frozen pages with multi-stream SSDs. In: Proceedings of the 15th International Workshop on Data Management on New Hardware, pp. 1-3 (2019)
11. Park, J. H., Choi, S., Oh, G., Lee, S. W: Gather Interface for Freezing Pages in Flash Storage. IEEE Access, 9, 102542-102548 (2021)
12. Bjørling, M., Aghayev, A., Holmberg, H., Ramesh, A., Le Moal, D., Ganger, G., et al.: ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. In: 2021 USENIX Annual Technical Conference (USENIX ATC 2021), pp. 689-703 (2021)
13. Ultrastar DC ZN540 from Western Digital, https://www.westerndigital.com/en-ae/products/internal-drives/ultrastar-dc-zn540-nvme-ssd?sku=0TS2096, last accessed 2024/11/24
14. Bjørling, M., Gonzalez, J., Bonnet, P.: LightNVM: The linux Open-ChannelSSD subsystem. In: 15th USENIX Conference on File and Storage Technologies (FAST 17), pp. 359-374 (2017)
15. Lee, C., Sim, D., Hwang, J., Cho, S.: F2FS: A new file system for flash storage. In: 13th USENIX Conference on File and Storage Technologies (FAST 15), pp. 273-286 (2015)
16. Zhang, Y., Yao, T., Wan, J., Xie, C.: Building GC-free key-value store on HM-SMR drives with ZoneFS. ACM Transactions on Storage (TOS) 18(3), 1-23 (2022)