applied sciences

MDPI

Article

Balloon: An Elastic Data Management Strategy for Interlaced Magnetic Recording

Chi Zhang [1], Song Liu [1,*], Fangxing Yu [1], Menghan Li [1], Wei Tang [2], Fei Liu [2] and Weiguo Wu [1,*]

[1] School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; chi.zhang@stu.xjtu.edu.cn (C.Z.); fangxingyu@stu.xjtu.edu.cn (F.Y.); lmh13052690592@stu.xjtu.edu.cn (M.L.)
[2] ByteDance US Infrastructure System Lab, ByteDance Inc., Mountain View, CA 94041, USA; weitang.0101@bytedance.com (W.T.); fei.liu@bytedance.com (F.L.)
* Correspondence: liusong@xjtu.edu.cn (S.L.); wgwu@xjtu.edu.cn (W.W.)

Abstract: Recently, the emerging technology known as Interlaced Magnetic Recording (IMR) has been receiving widespread attention from both industry and academia. IMR-based disks incorporate interlaced track layouts and energy-assisted techniques to dramatically increase areal densities. The interlaced track layout means that in-place updates to the bottom track require rewriting the adjacent top track to ensure data consistency. However, at high disk utilization, frequent track rewrites degrade disk performance. To address this problem, we propose a solution called Balloon to reduce the frequency of track rewrites. First, an adaptive write interference data placement policy is introduced, which judiciously places data on tracks with low rewrite probability to avoid unnecessary rewrites. Next, an on-demand data shuffling mechanism is designed to reduce user-requests write latency by implicitly migrating data and promptly swapping tracks with high update block coverage to the top track. Finally, a write-interference-free persistent buffer design is proposed. This design dynamically adjusts buffer admission constraints and selectively evicts data blocks to improve the cooperation between data placement and data shuffling. Evaluation results show that Balloon significantly improves the write performance of IMR-based disks at medium and high utilization compared with state-of-the-art studies.

Keywords: interlaced magnetic recording (IMR); hard disks; data layout; data shuffling; interlaced translation layer (ITL); storage management

check for updates

Citation: Zhang, C.; Liu, S.; Yu, F.; Li, M.; Tang, W.; Liu, F.; Wu, W. Balloon: An Elastic Data Management Strategy for Interlaced Magnetic Recording. *Appl. Sci.* **2023**, *13*, 9767. https://doi.org/10.3390/app13179767

Academic Editors: Hyokyung Bahn, Sungyong Ahn and Kyungwoon Cho

Received: 2 August 2023
Revised: 26 August 2023
Accepted: 27 August 2023
Published: 29 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the advent of new technologies such as cloud storage, scientific computing, medical imaging, and social networking, the creation of digital content is exploding, driving the need for cost-effective, high-capacity storage in the modern data center [1–3]. Conventional Magnetic Recording (CMR) disks, known for their affordability and reliability, have been fundamental in constructing modern storage systems. However, their potential to achieve higher areal density gains through reducing bit size has seemingly reached its limits due to the superparamagnetic effect (SPE) [4]. In response to this challenge, various magnetic recording technologies that utilize energy-assisted and optimized track layouts have been proposed to overcome the existing dilemma [5–8].

Among these candidates, Shingled Magnetic Recording (SMR) and Interlaced Magnetic Recording (IMR) technologies are considered to be cost-effective and have therefore gained widespread attention in recent years [9]. CMR disk (Figure 1a) tracks are separated by guard spaces, allowing random writes without affecting adjacent tracks. In contrast, the SMR disk (Figure 1b) reduces track pitch by enlarging the write heads and subtly overlapping the tracks like shingles to achieve higher areal densities. However, this gain in capacity comes at the cost of introducing a significant amount of extra I/Os, as random writes to any track require time-consuming cascading track rewrites to maintain the integrity of

adjacent valid data. Another promising successor to CMR, IMR, uses both heat-assist technology and an interlaced track layout to achieve capacity gains [8]. As shown in Figure 1c, an IMR disk conceptually divides tracks into the top and bottom tracks. Each narrower top track (smaller track pitch) is interleaved with two wider bottom tracks (higher line density). Writing to the bottom track requires stronger laser power than the top track, which introduces write interference that corrupts valid data in the top track. Consequently, writing to the bottom track requires rewriting two adjacent top tracks, a process known as read–modify–write (RMW). Although the write penalties of IMR are obviously lower than those of SMR, frequent RMWs degrade disk performance as space utilization (allocated track ratio) increases, limiting their use at scale [10,11].

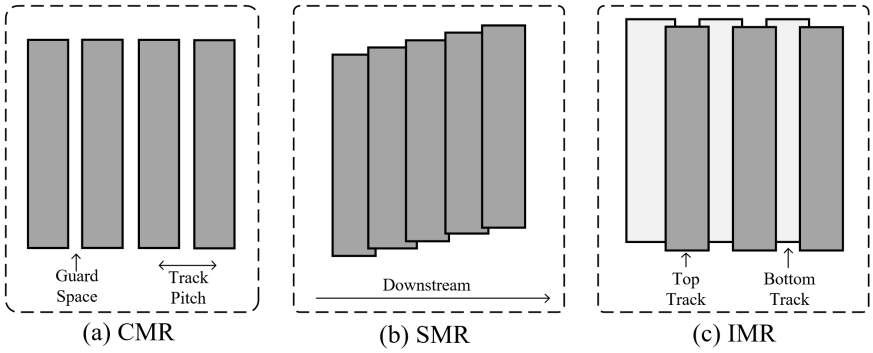


Figure 1. Track layout comparison for CMR, SMR, and IMR.

To date, there have been several efforts from academia and industry to reduce the frequency of track rewrites. These efforts mainly focus on optimizing data placement and data shuffling techniques. Data placement policies that take into account the write interference effect, statically or dynamically allocate tracks for incoming data to minimize unnecessary rewrite overhead. For example, a typical three-phase write management method [12,13] places incoming new data evenly across free tracks based on space utilization to delay the onset of RMW as much as possible. On this basis, Wu et al. [14] proposed a zigzag track allocation policy to improve the spatial locality of logical data blocks, resulting in reduced seek time. Compared with the static data placement policy, the dynamic data placement policy further reduces the track rewrite frequency by considering the track hotness. For example, during the top track allocation phase, MAGIC [10] attempts to place data on the free top track adjacent to the cold (infrequently updated) bottom track and redirects write interference data (if there is valid data on the top tracks adjacent to the bottom tracks, writing to these bottom tracks will cause write interference. Therefore, the term "write interference data" is used to describe data written to the bottom tracks that cause such interference) to a persistent buffer inside the disk, effectively reducing the introduction of track rewrite overhead. In practice, the top track can be written freely without introducing rewrite overhead. To take advantage of this property, data shuffling policies periodically or actively swap hot (frequently updated) data from the bottom track with cold data from the top track, at either track-level or block-level granularity. For example, Hajkazemi et al. [15] proposed three track-level data shuffling policies that minimize rewrite overhead by periodically moving bottom hot tracks to a persistent buffer or top tracks. Similarly, MOM [11] reduces extra I/Os by actively moving data to be rewritten to the free top track instead of backing up and restoring the data. Recently, Wu et al. [16] introduced two block-level data shuffling policies to improve the efficiency of data migration. The block-level data shuffling policy more accurately identifies hot data blocks, thereby improving the efficiency of each swap. In fact, data placement and data shuffling reduce the frequency of track rewrites from different perspectives; it is practical to use them together to achieve greater performance gains [10,14,16]. Nevertheless, existing research lacks a comprehensive solution for efficiently combining data placement and data shuffling. In particular, there is

a non-negligible performance gap between IMR disks and CMR disks under high space utilization, which greatly affects the practical use of IMR disks.

To mitigate the performance degradation caused by track rewrites, this paper presents an elastic data management strategy called Balloon, which is a device-level solution similar to the flash translation layer and the shingled translation layer. The principle behind this strategy is to proactively allocate incoming write interference data to the free top track. This not only eliminates the rewrite overhead of direct read–modify–write (RMW) operations but also enables low-cost data shuffling using the buffered data in the top track at no additional cost. As a result, the strategy uses on-demand data shuffling to relocate hot and cold data to specific regions, minimizing the frequency of track rewrites and the impact of data migration on user requests. The contributions of this research can be summarized as follows:

- An adaptive write interference data placement policy is proposed that reduces unnecessary rewrite overhead by judiciously placing data in tracks with low rewrite probability and opportunistically using the free top track to buffer write interference data. Meanwhile, a lightweight Bloom-filter-based hotness scoring mechanism for tracks is established to better balance the accuracy and memory overhead of track-level hotness identification.
- An on-demand data shuffling policy is introduced to further improve data placement efficiency. During data shuffling, hot tracks with low update block coverage are relocated to reserved regions of the disk to minimize the frequency of track rewrites, while tracks with high update block coverage are placed in the top region promptly to reduce seek time for subsequent update requests.
- To boost the collaborative efficiency of data placement and data shuffling, a write-interference-free persistent buffer is introduced. The persistent buffer dynamically adjusts admission constraints based on space utilization, thereby reducing the cleanup frequency. Moreover, it selectively evicts data blocks to minimize the frequency of track rewrites.
- The results of our evaluation show that the proposed strategy improves IMR write performance by up to 68.7% and 78.8% on average at medium and high space utilization, respectively, compared with state-of-the-art studies.

The rest of this paper is organized as follows. Section 2 describes the background and related work. The motivation is then presented in Section 3. Details of the Balloon design and implementation instructions are given in Section 4. Evaluation results and analysis are given in Section 5. Finally, Section 6 summarizes the study.

2. Background and Related Work

2.1. Background

Recently, various types of SMR disks have gradually been emerging in data centers and cloud storage providers to meet the demand for low-cost, high-volume storage [2,3]. However, the random writes of SMR disks suffer from significant write penalties, leading to their limited application scenarios and preventing them from fundamentally replacing CMR disks [10]. As an alternative to CMR disks, IMR disks are starting to gain attention with their higher capacity and lower write penalty. To achieve a more favorable tradeoff between in-place updates and areal density, the tracks of an IMR disk are logically divided into an equal number of top and bottom tracks, and each bottom track overlaps only two adjacent top tracks. With the benefit of energy-assisted technology, IMR-disk-based track layouts can achieve much higher areal densities than CMR disks and slightly higher areal densities than SMR disks. According to existing research, the bottom track provides more than 10% [15] or 27% [10,14,16] higher linear density than the top track. In practice, the capacity gap between the top and bottom tracks is attributed to the difference in laser power required to write data to them. The consequence of this is that the top track can be written freely, while writes to the bottom track destroy valid data in the adjacent top

track. Therefore, IMR disks introduce an RMW operation similar to that of SMR disks to ensure crash consistency. As shown in Figure 2, when the write request for bottom track 2 arrives, the disk first copies the valid data from top tracks 1 and 3 to the backup region of the disk, then writes the incoming data to bottom track 2, and finally migrates the backup data to the native location, which requires a total of three writes and two reads. Despite the extra I/Os introduced by the bottom track writes, the interlaced track layout allows the IMR to suffer from a much lower rewrite overhead than the SMR. Note that the non-overlapping region with the top track is sufficient to read valid data in the bottom track, leaving the read operation on the bottom track unrestricted. In addition, real-time data overwrite techniques and lazy data overwrite techniques [15] are used to overwrite data within the backup region to prevent the disk from restoring outdated backup data to its native location when the system is rebooted.

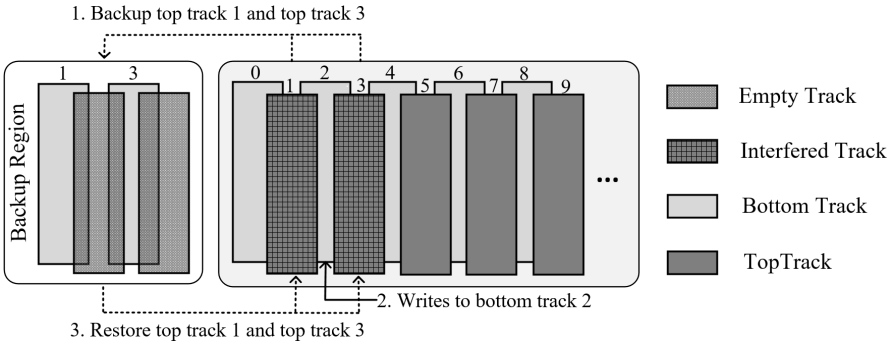


Figure 2. Illustration of the track rewrite handling.

2.2. Related Work

In terms of system implications and algorithms, previous studies have primarily focused on ways to minimize the occurrence of time-consuming RMW operations through data placement and data shuffling strategies. Data placement policies play a key role in allocating tracks to accommodate incoming data, with the goal of avoiding unnecessary track rewrites. Typically, data placement policies include two approaches: static track allocation based on space usage and dynamic track allocation based on track hotness. Classic static placement policies include the two-phase track write management method and the three-phase write management method [12,13]. In both policies, the bottom track is sequentially allocated when space utilization is below 50%. During this phase, all writes to the bottom track do not trigger RMW operations. When space utilization exceeds 50% (when top track allocation begins), the two-phase write management policy continues sequential allocation until the top tracks are exhausted, while the three-phase write management policy uses a jump allocation strategy. For example, after top track 1 is allocated, the next allocation is to top track 5 instead of top track 3. Consequently, the three-phase write management method reduces rewrite overhead in the 50% to 75% space utilization range, because at most one top track is involved in each rewrite. After space utilization exceeds 75%, the three-phase write management method allocates the remaining top tracks sequentially. Building on the three-phase write management method, Z-Alloc [14] uses zigzag allocation instead of radial allocation to improve the spatial locality of logical data blocks by reversing the allocation direction during the second phase. However, the frequency of track rewrites is not only determined by space utilization; the hotness of adjacent tracks is also a critical factor. Therefore, Liang et al. [10] proposed a bottom-up track allocation policy to strike a balance between logical data block sequentiality and track rewrite overhead. On the one hand, the logical data block is placed on the track adjacent to its physical location, which reduces the seek overhead of sequential I/O. On the other hand, each track is assigned a track update frequency counter that helps the disk place incoming data as close as possible to the top track with the lowest rewrite probability.

Data shuffling aims to achieve the desirable hot and cold data layout by migrating hot data to the top tracks where it can be freely written. To reduce the mapping overhead of redirected data, Hajkazemi et al. [15] proposed three techniques for data shuffling at the track level, namely selective track caching, track flipping, and dynamic track mapping. The core idea of these techniques is to reduce the introduction of track rewrites overhead by migrating the hot bottom track to the freely written top tracks or reserved region within the disk. However, these techniques either make poor use of the space available for freely written tracks or ignore the hotness of the adjacent bottom track, which limits their performance gain for IMR. In addition, to minimize the introduction of extra I/Os, MOM [11] suggests moving top tracks affected by write interference to other free top tracks with low rewrite probability instead of the typical RMW operation. However, this method retriggers the RMW operation when the disk is full because there are not enough free tracks to accommodate the moving data. As a typical representative of block-based data shuffling, Wu et al. [16] proposed Top-Buffer and Block-Swap, which can not only use unallocated tracks to buffer write-interfering data blocks but also actively swap eligible hot blocks to the top region, effectively improving the efficiency of data shuffling. However, with Top-Buffer, the buffer track contains data blocks from multiple bottom tracks, resulting in an unpredictable duration of data migration. In addition, Block-Swap potentially reduces the spatial locality of data blocks, and a single I/O request may be split into multiple subrequests, significantly increasing the latency of user requests. Unlike Top-Buffer, the unique feature of our proposed strategy is that top tracks acting as buffers can only accept data from the same target track. In this way, costless data migration can be achieved after all data in the original track are invalidated, thus reducing the overhead of explicit data shuffling.

3. Motivation

Compared with existing studies, our focus is on enhancing the write performance of IMR disks with high space utilization. To achieve this goal, the proposed strategy utilizes disk resources as much as possible to reduce the frequency of track rewrites and improve the efficiency of the cooperation between data placement and data shuffling.

In real-world workloads, new write requests and update write requests alternate on disk, meaning that free tracks are not immediately exhausted. To better illustrate this phenomenon, we analyzed the block-level traces obtained from Microsoft Research (MSR) Cambridge [17,18] and selected four traces, namely prn_0, src2_2, rsrch_0, and stg_0. These traces are write-intensive workloads, with the key difference being that the first two are dominated by spatial locality, while the latter two are dominated by temporal locality. In addition, since all existing IMR studies use a track-level data placement method (each logical track is mapped to a unique physical track, and incoming data are placed at the appropriate location on the physical track according to its offset within the logical track), we used the typical three-phase write management method as the baseline policy for data placement. Detailed information about the workload and experimental platform characterization can be found in Section 5.1. As shown in Figure 3, the temporal-locality-dominated workloads, rsrch_0 and stg_0, trigger top-track allocation when write progress reaches 6% and 9%, respectively. In contrast, the spatial-locality-dominated workloads prn_0 and src2_2 have slower track consumption rates, with top track allocation triggered only when write progress is 66% and 32%, respectively. Although rsrch_0 and stg_0 initiate top-track allocation early in the workload, the rate of free track allocation gradually slows as write progress increases. In summary, regardless of the rate of free track consumption at each stage of the workload, there are always free tracks available on the disk until the write progress reaches 100%.

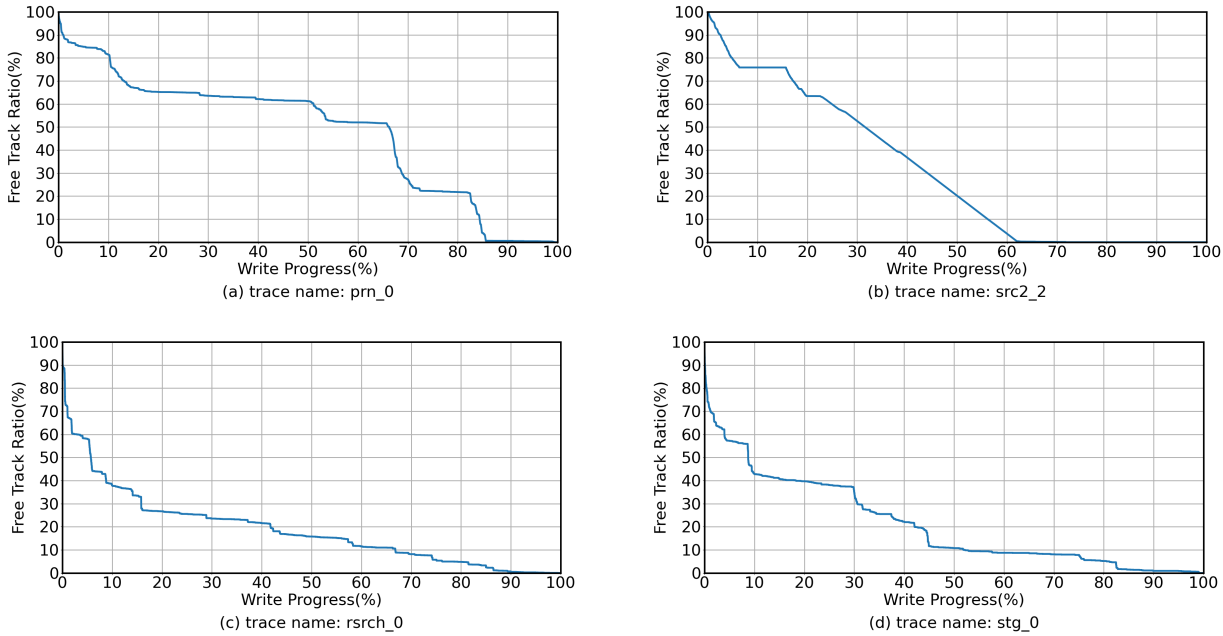


Figure 3. The track allocation rate for different workloads: The X-axis indicates the the percentage of completed write requests and the Y-axis shows the percentage of free tracks.

Inspired by the above observation, if free top tracks can be used to accommodate write interference data, then track rewrites can be efficiently suppressed. Therefore, the key to reducing rewrite overhead is to find a rational approach to using the free top track as a buffer track. However, this is not an easy task, and the challenge lies in the following aspects: (1) how to select the right candidate to act as a buffer track without accelerating the consumption of free tracks; (2) how to use buffer tracks to facilitate low-cost data shuffling; and (3) how to handle track rewrites when there are no reclaimable top tracks to act as buffer tracks.

4. Balloon Design

4.1. Overview

In this section, the Balloon system architecture and the main modules it contains are introduced. As shown in Figure 4, to avoid exposing complex data write constraints to the host system, Balloon is implemented as a translation layer in the internal firmware of the disk. Specifically, Balloon translates the logical block addresses of incoming I/O requests into the physical block addresses of the disk and performs data migration within the disk promptly to reduce the frequency of track rewrites. In addition, Balloon is responsible for ensuring metadata reliability. To achieve this, Balloon consists of three key modules: adaptive write interference data placement, on-demand data shuffling, and a write-interference-free persistent buffer (see Sections 4.2–4.4, respectively).

4.2. Adaptive Write Interference Data Placement

4.2.1. Track Allocation

As described in Section 2.1, the tracks on an IMR disk are conceptually divided into bottom tracks and top tracks. To simplify the discussion in the subsequent sections, the collections of the bottom tracks are referred to as bottom regions and vice versa for the top regions.

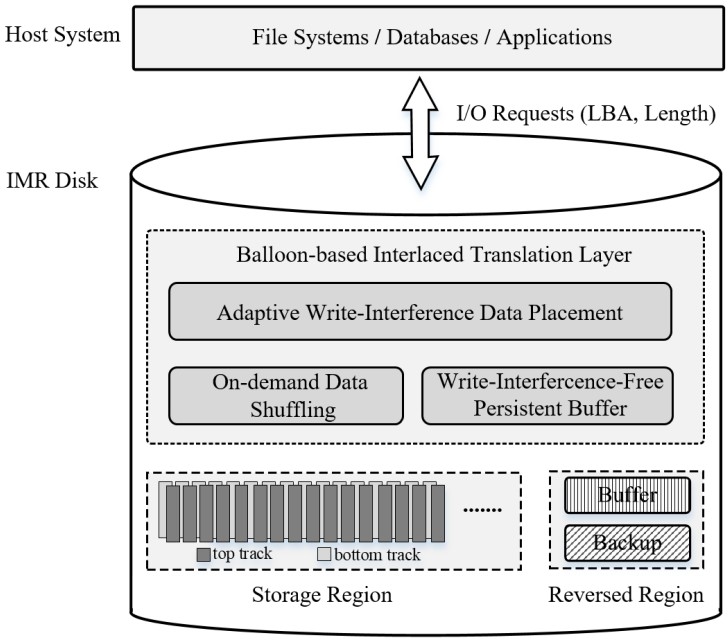


Figure 4. The overview of the proposed Balloon translation layer.

The goal of the adaptive write interference data placement module is to improve placement efficiency by dynamically placing data on the specified track based on seek distance or track hotness. To prevent the unnecessary introduction of track rewrites, the allocation of tracks in the bottom region takes precedence. Initially, all unallocated tracks reside in the free track pool. As long as there are free tracks in the bottom region, any new incoming data are placed on these tracks. To minimize seek latency, the track allocator searches for the nearest free bottom track in the inside diameter (ID)/outside diameter (OD) direction, starting with the track accessed by the last I/O request. As shown in Figure 5a, there are no valid data that exist above the bottom track at this stage; so, it can be written freely without incurring additional rewrite overhead.

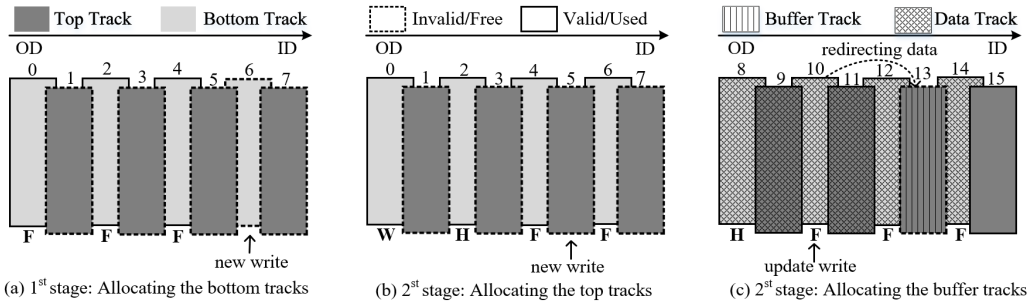


Figure 5. Illustration of the adaptive write interference data placement: The track allocator determines which tracks to place data on based on space utilization and the write request type, where *F*, *W*, and *H* denote track hotness, i.e., frozen, warm, and hot, respectively.

Once the free tracks in the bottom region are exhausted, track allocation immediately switches from the bottom allocation phase to the top allocation phase. From then on, writes to the bottom track with write interference will trigger track rewrites. To reduce the frequency of track rewrites, dynamic allocation is proposed to guide the allocation of tracks in the top region. Specifically, the track allocator selects top tracks for incoming data based on the hotness of the bottom track, thus minimizing the probability that it is involved in rewrites. The evaluation of track hotness is described in Section 4.2.2. As shown in Figure 5b, when new data arrive, the track allocator searches for eligible candidates along the ID/OD direction. For free top track 1, it is not an ideal candidate, because the

neighboring hot bottom track 0 and warm bottom track 2 have a higher probability of engaging it in track rewrites. In contrast, frozen tracks 4 and 6 indicate that they have not been updated recently, making it unlikely that the adjacent top track 5 is involved in track rewrites. Therefore, track 5 is treated as a candidate for allocation. However, as space utilization gradually increases, more and more top tracks are affected by write interference, which inevitably triggers track rewrites. To amortize and reduce the extra I/O generated by rewrites, the track allocator selects the eligible free top track to act as a temporary buffer for the target track to accommodate the write interference data. Therefore, from a functional perspective, tracks are further divided into data tracks and buffer tracks. It should be noted that since block-level I/O typically exhibits spatial locality, each buffer track is used exclusively. In fact, not only does this track-level caching maintain logical data block continuity, but the redirected data blocks do not incur costly metadata management overhead. As shown in Figure 5c, both the top track 9 and the top track 11 adjacent to the bottom track 10 contain valid data. Upon the arrival of the update request for bottom track 10, the track allocator selects the free top track 13 from the top region as its buffer track instead of the direct RMW operation. Thereafter, any data block that is committed to the bottom track 10 is redirected to the same offset position within the buffer track until the on-demand data shuffling is triggered.

4.2.2. Lightweight Track Hotness Scoring

The existing methods for evaluating track hotness capture track hotness by setting a cumulative update counter for each track [10,11,15]. For terabyte-capacity IMR disks, this method of identifying hot and cold data imposes significant computational and storage overhead. To trade off identification accuracy and memory overhead, Balloon uses multiple Bloom filters instead of a cumulative update counter to evaluate track hotness. However, traditional multiple Bloom filters are designed to evaluate the hotness of data blocks in limited-capacity flash memory, and this fine-grained decision-making approach is not suitable for large-capacity IMR disks, where the track allocator is expected to know hotness information at the track level rather than the block level. For this reason, the coarse-grained track-level decision replaces the fine-grained block-level decision.

Figure 6 shows the proposed track-level multiple Bloom filter (TMBF), which consists of M hash functions and N-independent Bloom filters (BFs), where each BF contains P bits to store M hash values. When an update request arrives, the target track number is entered into the TMBF, and then the hash value generated by each hash function is inserted into the corresponding bit in the specified BF. If the bit of the current BF is "positive" (i.e., the corresponding bit of the track in the BF is 1), the TMBF selects the next BF in a round-robin fashion. As shown in Figure 6, for track 2, the corresponding bit in BF_1 is positive (indicated by the green rectangle) and BF_2 is not yet set, so BF_2 is selected to set the bits. In practice, the number of consecutive positive results in the TMBF indicates the frequency of track updates. Since the number of BFs is limited, a freezing method [19,20] is used to prevent bit overflow if the corresponding bits in all BFs are positive. In addition, the TMBF periodically zeroes all bits of the oldest BF in a polled fashion to capture the recency of track hotness. To further improve the classification accuracy, the hotness of tracks is classified as frozen, warm, and hot. Specifically, in the TMBF, a frozen track has no "positive" results; a warm track has several consecutive "positive" results between (0, N); and a hot track has at least N positive results. The reason for this is that our scheme approximates the hotness of tracks by checking whether the track has been updated rather than the exact number of updates. Therefore, for a frozen track that has never been updated, we can assume that it is unlikely to be updated.

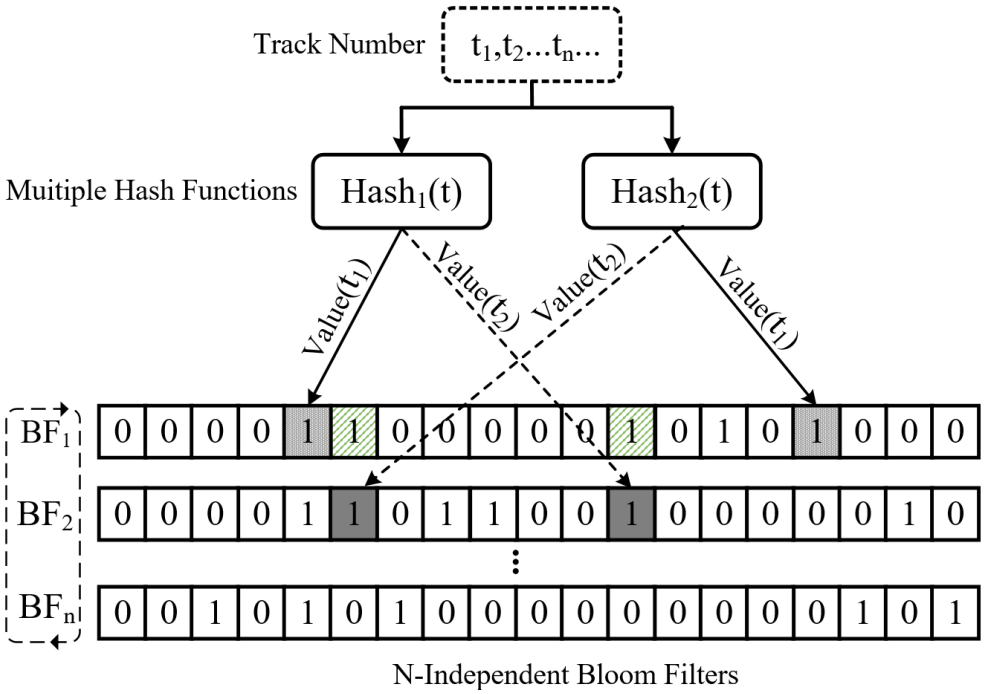


Figure 6. A track-level multiple-Bloom-filter-based framework for track hotness scoring.

4.3. On-Demand Data Shuffling

After switching to the top allocation phase, some free top tracks are used as temporary buffers for bottom tracks with write interference, which accelerates the exhaustion of the free top tracks. When the free top track is used up, the track merge/swap operation is enabled to reclaim the free top track. In the Balloon, the track relocator determines its target storage locations based on the hotness of the tracks. In essence, the track relocator tries to place cold and hot data in the bottom and top tracks, respectively, to achieve the desirable data layout. To reduce the frequency of track rewrites, the track relocator preferentially searches for cold tracks as candidates. Typically, the selected candidate is either a data track or a buffer track. For the former, the track relocator uses a swap operation to relocate it to any free track in the bottom region. As for the latter, the track relocator merges it directly into the data track to which it is associated with. The assumption here is that an updated track is more likely to be updated again than an un-updated track, so the data track is given a higher merge/swap priority. In addition, once the bottom track is associated with the buffer track, subsequent writes are directed to the buffer track, which can invalidate the original data in the data track. As shown in Figure 7, when there are no valid data in the bottom data track 2, the buffer track 5 is automatically converted to a data track, and the previous data bottom track 2 becomes free again. This implicit merge operation can migrate hot and cold data without introducing extra I/Os, significantly reducing the cost of data shuffling. It is essential to emphasize that before swapping a data track into the bottom region, the track relocator first checks for the availability of a free bottom track to accommodate its data. If no free bottom track is available, the track relocator will merge a hot data track from the bottom region into its buffer track, thereby creating a free track. For instance, the hot bottom track 22 is merged upward into buffer track 19, thus freeing up a usable bottom track 22. In contrast, if the candidate is a buffer track, the track relocator directly performs a downward merge directly without considering the availability of free tracks in the bottom region. For example, the candidate is buffer track 9, which is merged downward into its data track 14.

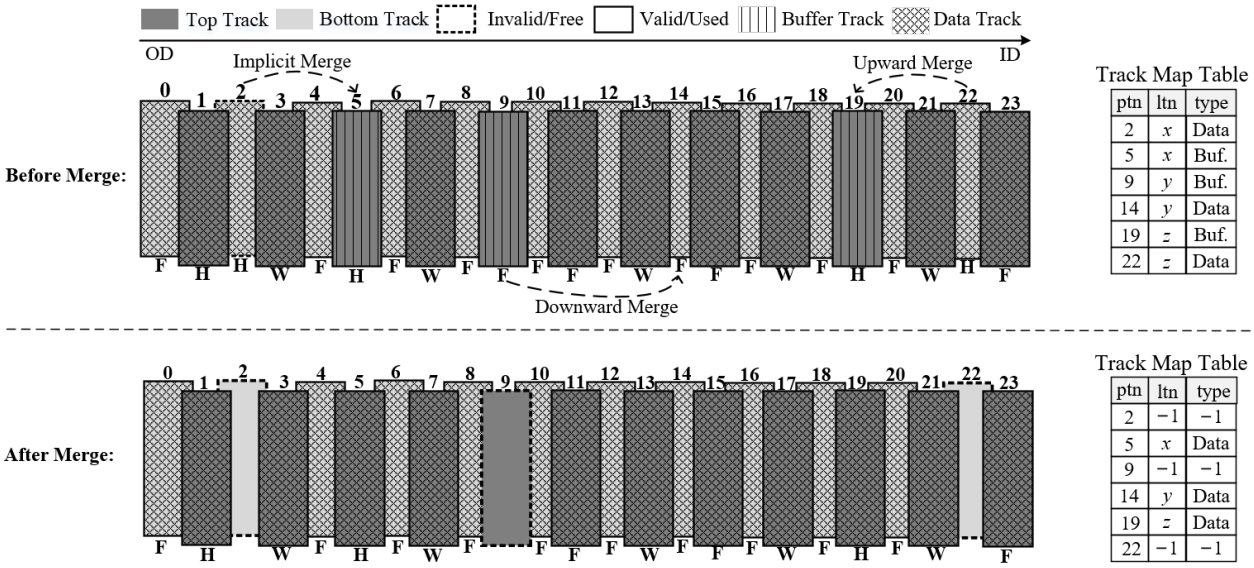


Figure 7. Illustration of the track merge operation, where x, y, and z represent the logical track numbers and F, W, and H denote track hotness, i.e., frozen, warm, and hot, respectively.

However, apart from implicit merges, both up/down merges and track swaps introduce non-negligible extra I/Os, which are costly. Specifically, a data track swap requires 4~8 extra I/Os, while a buffer track merge requires at least 3 extra I/Os. If there are no cold tracks to act as candidates, there is a high probability that swapping warm/hot tracks to the bottom region will increase the frequency of track rewrites, thus deteriorating disk performance. To solve this problem, blocks of warm/hot tracks are relocated to a reserved region on the disk to avoid track rewrites. Reserved region management is described in Section 4.4. Also, in real workloads, the distribution of updated blocks within a track is not uniform; some tracks have a small update range (few updated blocks), while others have a large update range (most updated blocks). Therefore, it is beneficial to reserve the tracks with large update ranges in the top region to better exploit the spatial locality of the tracks and reduce the write latency of user requests. To evaluate the update range of a track, a metric called update block coverage is introduced. In our design, a bit array is used to capture the distribution of updated data blocks within the track, and its size is equal to the number of blocks contained in the track. In summary, when there are only warm/hot tracks in the upper region, the track with the lowest update block coverage is selected as a candidate and swapped to the reserved region. As more and more new data arrive, the buffer tracks are gradually returned to the user to accommodate the new data. When the free tracks in the top region are exhausted and no buffer tracks exist, the track merge operation is disabled.

4.4. Write-Interference-Free Persistent Buffer

The persistent buffer is proposed to improve the efficiency of cooperation between data placement and data munging; so, effective management of data in the persistent buffer is critical to improving disk performance. As shown in Figure 8, the reserved region consists of a set of bottom tracks located on the OD of the disk, and the adjacent top tracks are intentionally left empty so that the bottom tracks can be written freely. For this reason, this reserved region is also called a write-interference-free persistent buffer (WIF-PB). To improve space utilization, WIF-PB is organized in a circular log structure and uses a block-level rather than a track-level approach to data management. New data blocks that arrive in the WIF-PB are appended to the end of the circular log to reduce write latency. However, the capacity of WIF-PB is limited, and the unrestricted admission of data blocks triggers frequent buffer cleanups, leading to disk performance jitter. Therefore, Balloon regulates the admission constraints of the WIF-PB based on top track usage to minimize

the cleanup overhead. Specifically, during the period when the track merge operation is enabled, if there are data blocks of the target track in the WIF-PB, all subsequent write requests for that track will be redirected to the WIF-PB. By doing so, it prevents the data blocks of the target track from being too widely scattered, thereby reducing the seek overhead of I/O requests. After the track merge operation is disabled, any write interference data can be redirected to the WIF-PB because there are no more free tracks in the top region to act as buffer tracks.

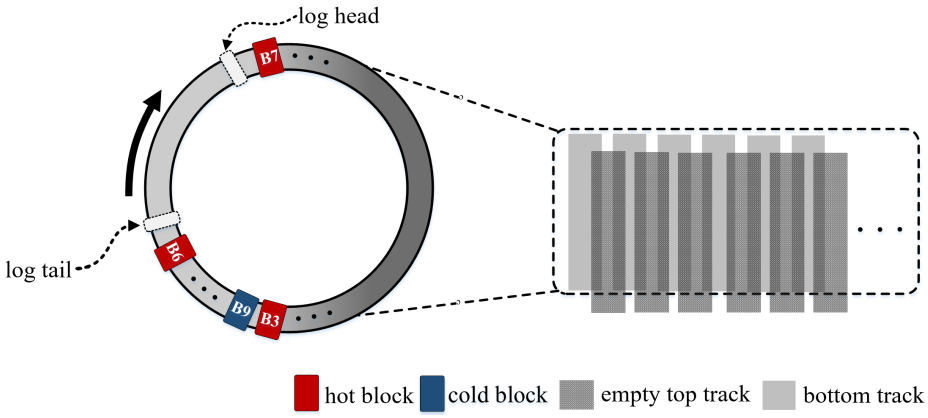


Figure 8. Architecture for the write-interference-free persistent buffer.

When there is no space in the WIF-PB for append write, Balloon evicts the data block at the end of the circular log to the bottom region as a victim. To better amortize the data blocks relocation overhead, other buffer blocks subordinate to the same logical track as the victim are also written back to the bottom region. Although the above approach effectively reduces the cleanup frequency, the FIFO-based cleanup policy inevitably evicts hot data blocks prematurely. The RMW operations introduced by hot data going back and forth between the WIF-PB and the bottom track cause disk performance jitter. To address this problem, Balloon uses a requeuing technique to move hot data blocks in the victim to the log tail. It also introduces a simple but effective method to minimize the overhead of identifying hot data blocks. Specifically, each block is flagged with an initial value of FALSE. Once they are updated in the WIF-PB, this flag is set to TRUE. As a result, only blocks with a TRUE flag can be requeued, and all other blocks are simply written back as cold data. To reduce the overhead of requeuing and to improve the accuracy of hot data identification, all flags are periodically cleared. Furthermore, as an optimization, we proposed that if the update block coverage of the victim track is higher than that of the track with the lowest update block coverage in the top region, the track relocator actively swaps them to achieve a potential gain. The benefits of this are twofold: (1) it improves the spatial locality of the data blocks in the track and (2) it eliminates the rewrite overhead caused by subsequent writes to the bottom track.

4.5. Implementation Description

4.5.1. I/O Request Processing Summary

Upon the arrival of a write request, the track relocator first checks to see if the top track needs to be reclaimed and performs the merge/swap operations if necessary. Next, the track relocator identifies the type of target track. If writes to the target track do not cause write interference, the write request is routed to the target track and processing is completed. Otherwise, the track allocator must further examine other constraints to determine the location to redirect the write request. Specifically, if the write request satisfies the admission constraints of the WIF-PB, then it is routed there. Otherwise, the track allocator needs to allocate a buffer track for the target track and place the data blocks in the buffer track according to their offsets in the target track. The processing flow for read

requests is straightforward, simply collecting data blocks from the buffer track (if it exists), the WIF-PB, and the target track, in that order.

4.5.2. Data Security and Crash Consistency

During the track merge/swap and WIF-PB cleanup processes, there is a possibility of a system crash resulting in data loss. To ensure data reliability, valid data in the tracks involved in data shuffling need to be backed up promptly. Specifically, the valid data and mapping information of the target track and the adjacent top track are copied to the backup region before the merge/swap operation is performed. Redundant backups are not required for the swapped top tracks, because even if the system crashes, valid data are still present in the original location. The process of merging the bottom track into the buffer track in the top region is similar to the process described above, except that the process does not affect adjacent tracks; so, only the data in the buffer track needs to be backed up. It is important to note that with both track swapping and track merging, the old metadata information is not discarded before the new mapping information is persisted. In this way, even if a system crash occurs during data shuffling, the correct metadata information can still be found and the system can be rolled back to its previous state. Ensuring data consistency during persistent buffer cleanup is widely discussed [21,22] as a standard technique in SMR disk research. Therefore, the same practice can be used to ensure data consistency during WIF-PB cleanup. In addition, if the additional cost is acceptable, using a small amount of flash or nonvolatile memory to act as a backup region can effectively reduce the read and write latency of backup data.

4.5.3. Metadata Overhead Analysis

For the proposed Balloon, maintaining metadata information at different granularities is essential. Based on a reference model of an 8 TB IMR disk and a 25 GB persistent buffer [10], with a track size of 2 MB and a physical block size of 4 KB, the detailed metadata overhead of the proposed strategy is described below.

To efficiently locate tracks in the top and bottom regions, a mapping table called TrackMapTable is used to record the logical-to-physical track relationship. Each entry in the table occupies 4 bytes, resulting in a total size of approximately 16 MB. Additionally, the FreeTrackTable is introduced to quickly find an available track. It is a 512 KB bitmap where each bit represents the status of a track, indicating whether it is available or not. At system startup, both the TrackMapTable and the FreeTrackTable are loaded into internal disk memory to speed up the query. To track the distribution of updated blocks in a track, the track allocator assigns an update block coverage recorder (a 64-byte bitmap) to each updated top track. Because the number of updated tracks varies with different workloads, the memory overhead for the update block coverage record is dynamic. To avoid unnecessary memory consumption, no more than half of the total number of top tracks are loaded into memory, which requires 64 MB of memory. The remaining updated block coverage information is stored on disk and loaded into memory as needed. For WIF-PB, each block-level mapped entry consumes 8 bytes, resulting in an approximate memory requirement of 50 MB. In addition, the proposed TMBF size can be configured to meet the QoS requirements of different users. In this paper, the TMBF values for N and P are set to 8 and 2048, respectively, resulting in a total memory overhead of 2 KB. In total, the proposed strategy uses up to 132 MB of memory at runtime, which is a reasonable overhead for high-capacity disks currently available in the market (such as typical SMR disks equipped with 256 MB DRAM) [10].

5. Performance Evaluation

5.1. Evaluation Setup

In this section, the effectiveness of the proposed Balloon translation layer is evaluated and the related evaluation results are reported. Since IMR is a very new technology, there are no IMR disks in the market yet. Therefore, we simulate the characteristics of an IMR

disk based on a real CMR disk (model ST500DM002 [23]). The track size of the simulated IMR disk is set to 2 MB, which is set concerning recent studies [10,11,15,24]. The prototype system of the proposed strategy provides a block interface for the upper layer to replay I/O requests from block-level trace files. Specifically, all I/O requests are submitted to the simulated device through read and write functions provided by the open-source library libaio [25]. To improve the accuracy of the evaluation, all replayed read and write requests are direct I/O, bypassing the kernel buffer, and the on-disk cache is disabled. In addition, the capacity of the IMR disks was not set to a fixed value but determined by the unique track count written in the currently replayed trace to evaluate the performance of Balloon and competitors under different space utilization. All evaluations are performed on an AMD workstation with EPYC 7302 CPU 3.0 GHz and 128 GB RAM. The operating system is Ubuntu 20.04.1 based on the Linux 5.15.0 kernel.

Our goal is to narrow the performance gap between IMR disks and CMR disks to make them suitable for traditional server application scenarios. To achieve this, we selected a set of write-intensive tracks from the MSR collection [17,18], since read-intensive workloads may not result in track rewrites. The write requests in the selected trace exhibit various characteristics, including temporal and spatial locality, as well as random and sequential nature. Table 1 provides details about these characteristics. Throughout this study, our focus has been on evaluating the performance of the proposed strategy under high space utilization. Therefore, the number of available tracks is dynamically adjusted based on the footprint (total number of unique tracks written) of the selected trace. For example, at 100% space utilization, the total number of available tracks is equal to the total number of unique tracks written in the selected trace. Similarly, at 50% space utilization, the total number of available tracks is twice the number of unique tracks written.

Table 1. Basic statistics of the selected workloads.

Trace Name	Total Requests	Write Req. (%)	Footprint	Description
prn_0	5,585,886	89.2	8084	Seq&Rnd
src2_2	1,156,885	69.7	15,162	Seq
rsrch_0	1,433,655	91.8	2280	Cold&Hot
stg_0	2,030,915	77.5	1830	Hot

In addition to the CMR disks used as a baseline, state-of-the-art ITL designs are carefully selected as Balloon competitors. These competitors include three-phase write management (Seagate) [12,13], selective track caching (STC) [15], dynamic track mapping (DTM) [15], MOM [11], and MAGIC [10], and their fundamentals are described in the related work in Section 2. We also included a variant of Balloon in the evaluation, called Balloon-MS, which disables the WIF-PB and allows only merge and swap operations. This variant helps us observe the extent of performance improvement on IMR disks with limited disk resources. To ensure a fair evaluation, the parameters of all competitors are strictly set according to the source paper. Specifically, the update frequency of the selective track caching and dynamic track mapping policies is set to 20 K, with the persistent buffer size of the former set to 1% of the disk capacity (see the settings of the latest SMR disks), similar to the settings of the Balloon and MAGIC, while the latter has a hot/cold threshold of 50 and includes 256 pseudotracks per zone. For MAGIC, we use the default region size of 64 MB as suggested in the original paper. In Balloon, referencing studies [19,20], the TMBF is configured with N = 8 and p = 2048, and the oldest BF is reset every 256 update requests for different tracks to decay the records in the TMBF. All these settings ensure a fair and comprehensive comparison between the ITL designs.

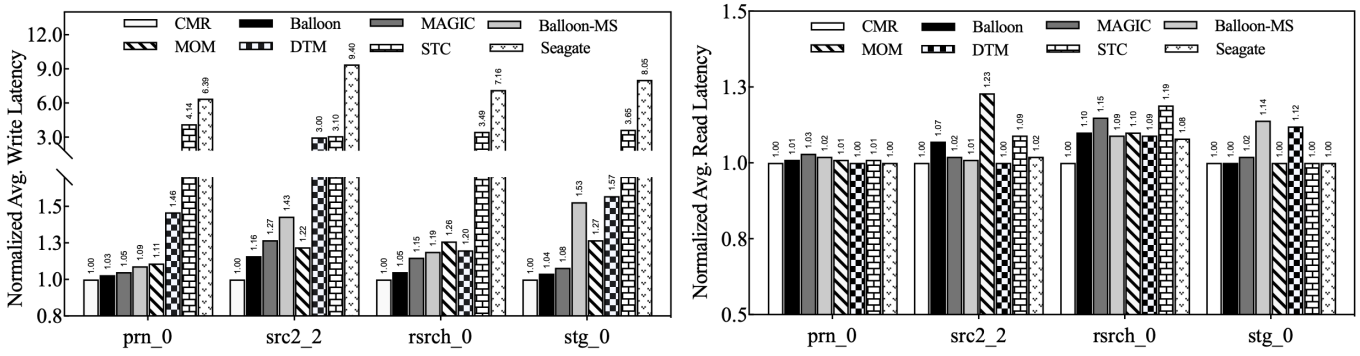
5.2. I/O Performance Evaluation

In general, the write performance of IMR disks exhibits fluctuations as the space utilization increases due to the increasing number of tracks triggering RMW operations. To evaluate the effectiveness of the proposed Balloon strategy, we evaluate the performance

of IMR disks at 80% and 99.99% utilization (considered as medium and high utilization, respectively). These scenarios help us understand how the strategy performs at write-intensive workloads. In our evaluation, we analyze in detail the improvements in write and read performance achieved by the proposed strategy. While the average I/O latency of a workload provides insight into the overall performance of the disk, it may not fully capture the true read/write performance. Therefore, we focus on understanding how Balloon improves the read and write performance of IMR disks.

5.2.1. Write Performance

Figure 9a shows the comprehensive write performance comparison between Balloon and its competitors at medium utilization. Notably, the write performance of Balloon-based IMR disks is consistently the closest to that of CMR disks across different workloads. The similarity in performance between Balloon and MAGIC can be attributed to two factors: First, during the top allocation phase, Balloon and MAGIC dynamically select free top tracks based on the hotness of the bottom tracks. When disk utilization is at a medium level, a number of free tracks are available, making it easy to place data on tracks with low rewrite probability. Second, both strategies utilize the persistent buffer to manage write interference data. At medium utilization, the persistent buffer experiences fewer writes, which reduces the impact of cleanup activities on write performance. Despite these similarities, Balloon still outperforms MAGIC with an average write performance improvement of 5.74%. As shown in Figure 10a, at high utilization rates, Balloon achieves a significant improvement in IMR disk write latency over its competitors, ranging from 6.3% to 92.4%. Of particular note is the 11.3% average improvement in write performance over MAGIC. The effectiveness of Balloon in improving write performance is attributed to its skillful use of the free top track as buffer tracks and its resilient persistent buffer admission and eviction constraints. In addition, Balloon's lightweight track hotness scoring mechanism periodically discards historical data to adapt to current workload access patterns. The above measures significantly reduce the frequency of track rewrites and the impact of buffer cleanup on write performance.

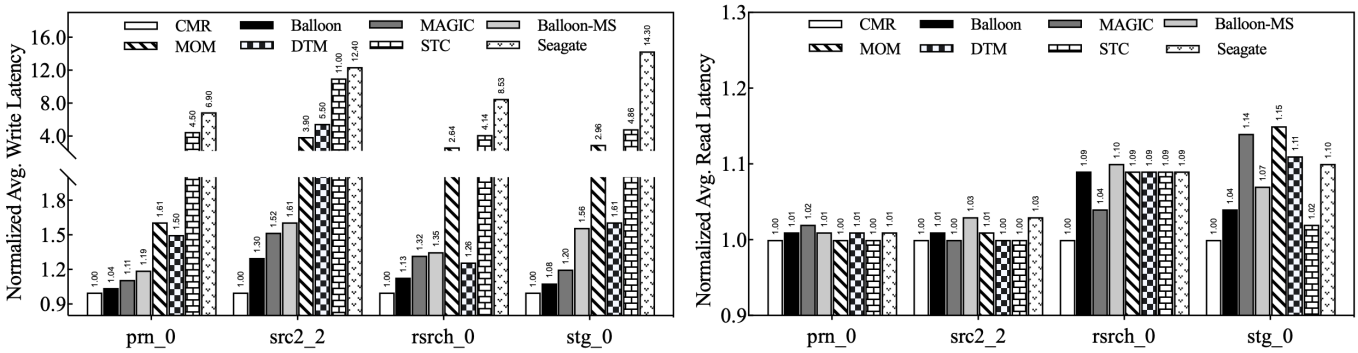


(a) Average write latency at 80% space utilization.

(b) Average read latency at 80% space utilization.

Figure 9. I/O performance of Balloon vs. competitors at 80% space utilization, where the X-axis represents different workloads and the Y-axis represents write latency, read latency, and average latency normalized to CMR.

In addition, we also focused on the performance gap between Balloon-MS and other competitors without a persistent buffer. As shown in Figure 9a, Balloon-MS outperforms its competitors under both prn_0 and rsrch_0, achieving an average write performance improvement of 45.9% and 38.9%, respectively, while its write latency is higher than MOM under src2_2 and stg_0. This result is due to the fact that Balloon-MS makes full use of the available tracks to act as buffer tracks, and reclaiming buffer tracks introduces some overhead.



(a) Average write latency at 99.99% space utilization.

(b) Average read latency at 99.99% space utilization.

Figure 10. I/O performance of Balloon vs. competitors at 99.99% space utilization, where the X-axis represents different workloads and the Y-axis represents write latency, read latency, and average latency normalized to CMR.

At high utilization rates, as shown in Figure 10a, Balloon-MS consistently outperforms MOM. It is noteworthy that both MOM and Balloon-MS resume RMW operations when the disk reaches full capacity, because there are no free tracks available to store the migrated data or act as buffer tracks. However, unlike MOM, which simply moves interfered data to other top tracks to minimize data rewrites, Balloon-MS proactively uses buffer tracks to migrate hot data to the top tracks, effectively reducing the frequency of track rewrites. This key advantage is responsible for the superior write performance of Balloon-MS over MOM in various high-load scenarios.

5.2.2. Read Performance

Figures 9b and 10b illustrate the differences in read performance between the evaluated strategies at medium and high utilization. Notably, the read latency is close for all strategies regardless of the utilization level. For DTM, STC, and Seagate, which use a one-to-one track mapping mechanism, the logical data block is concentrated on a specific physical track. As a result, their differences in read performance are caused by their respective data placement or data shuffling behaviors.

Distinctly from the aforementioned competitors, MAGIC and Balloon employ approaches that redirect write interference data to the persistent buffer or buffer track to avoid track rewrites. While this approach may intuitively result in slightly lower read performance compared with other competitors, as the block-level persistent buffer may split some read requests into multiple subrequests, the actual impact is mitigated by the specific design of MAGIC and Balloon. MAGIC uses the region mechanism to preserve logical data block sequentiality whenever possible, sacrificing spatial locality only at high utilization rates to reduce track rewrite overhead. Similarly, Balloon uses strict data placement constraints to improve the spatial locality of the logical data blocks. For example, write interference data, other than data tracks, can only be placed on one of the buffer tracks or the WIF-PB to prevent scattering of adjacent logical data blocks. These constraints effectively reduce the seek overhead of read requests and prevent significant read performance degradation.

5.3. Number of Extra I/Os

Extra I/O refers to track-level I/O caused by RMW operations or data shuffling apart from the user request. Therefore, the number of extra I/Os is a critical factor affecting write performance. Figure 11a,b shows the number of extra I/Os introduced by each strategy at medium and high utilization, respectively.

Seagate generates the highest number of extra I/Os in all scenarios because it only places incoming data on allocated tracks without using data shuffling. Conversely, other strategies significantly reduce the number of extra I/Os by periodically moving hot data to free-write regions, demonstrating the effectiveness of data shuffling. STC partially reduces track rewrites by placing hot tracks in the persistent buffer. However, the track-level data

organization approach does not fully utilize the limited capacity of the persistent buffer, resulting in limited gains from data shuffling. On the other hand, DTM allows arbitrary hot and cold tracks to be swapped within zones and moved more bottom tracks to free-write regions, resulting in significantly fewer extra I/Os than STC. MOM uses hotness-aware data movement and performs well at moderate utilization. However, at high utilization, its extra I/Os increases dramatically because its data shuffling is limited to the top region, leaving hot data in the bottom track unmoved to the top region. Once the free tracks are exhausted, the expensive RMW operation is reactivated, resulting in a significant amount of extra I/O. Both Balloon-MS and MOM opportunistically use free top tracks to reduce track rewrite overhead. However, Balloon-MS goes a step further with its track merge/swap operation, which enables cost-free data shuffling using buffer tracks, effectively suppressing the introduction of extra I/O. Equipped with a block-level persistent buffer, MAGIC introduces significantly fewer extra I/Os than any other strategy except Balloon. In particular, Balloon has the lowest number of extra I/Os in all scenarios, which contributes significantly to its remarkable improvement in write performance.

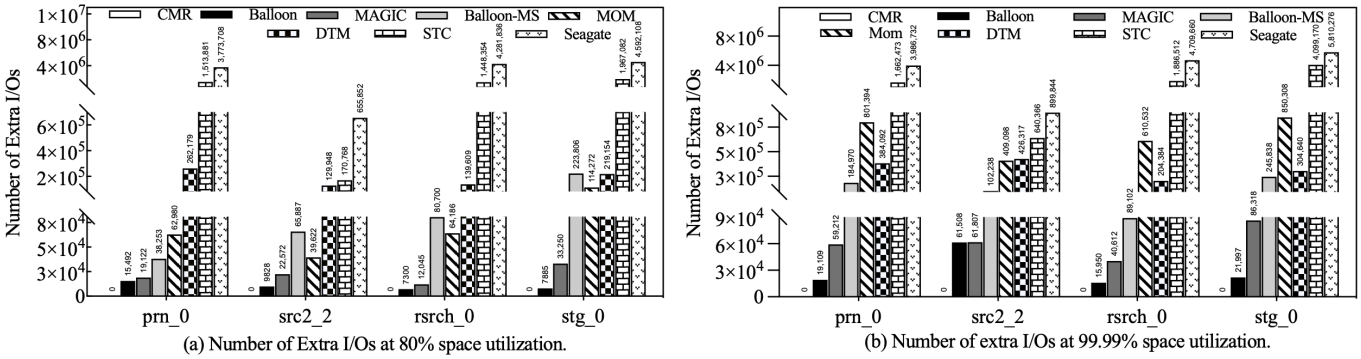


Figure 11. The total number of extra I/Os introduced by Balloon and competitors under different space utilization and workloads.

5.4. Persistent Buffer Cleanup Efficiency

Of all the strategies evaluated, Balloon and MAGIC stand out by introducing a persistent buffer to reduce the frequency of track rewrites. However, frequent cleanups degrade disk performance [21,22]. To further investigate the factors contributing to the performance gap between Balloon and MAGIC, this section focuses on evaluating the cleanup efficiency of their persistent buffers under varying space utilization and workloads. Therefore, the number of cleanups and the average size of the victims are used as performance metrics, which are commonly used in similar studies [26–29].

Figure 12a shows the buffer cleanup frequency of Balloon and MAGIC at different utilization and workloads. We can observe that both have lower cleanup frequencies at medium utilization than at high utilization. In particular, the cleanup frequency of Balloon is significantly lower than that of MAGIC in all scenarios. The reason for this is that Balloon allows only eligible data to enter the WIF-PB and swaps some buffer tracks from the top region directly to the bottom track, effectively increasing the efficiency of persistent buffer usage. In contrast, MAGIC indiscriminately directs write interference data to the persistent buffer, which results in frequent cleanups. Furthermore, Balloon takes proactive measures to retain hot data in the victims, preventing them from frequently moving in and out of the persistent buffer. This approach also contributes to a reduction in cleanup frequency. As another metric of cleanup efficiency, the average size of victims reflects the number of buffer blocks evicted per cleanup. The average size of the victims has a double meaning for evaluating the efficiency of each cleanup: first, the size of the victims somehow reflects the maximum spatial gain of the current cleanup operation; second, the size of the victims can reflect the effectiveness of Balloon in improving the spatial locality of the data. In general, the size of the victim is affected by a number of factors, including workload characteristics,

hot data identification mechanisms, and data management policies. Figure 12b shows the average size of victims evicted by Balloon and MAGIC for different utilization and workloads. Intuitively, the victim size of Balloon may not be as large as that of MAGIC, because it only redirects eligible data blocks to the persistent buffer. However, in most scenarios, the victim size of the Balloon is higher or close to that of MAGIC. Upon careful analysis, we find that most victims of the Balloon migrate from buffer tracks and tend to accumulate a significant number of data blocks before being swapped to the persistent buffer, potentially increasing the size of the victim. In a few cases, Balloon's victims are smaller than MAGIC's. The reason for the above result is that Balloon actively filters hot data from its victims, resulting in a reduction in victim size. In fact, Balloon reduces the cleaning frequency at the cost of reducing the size of the victims, which improves the efficiency of buffer cleaning in the long run. Therefore, the size of the victim is not a determining factor in the efficiency of the cleanup.

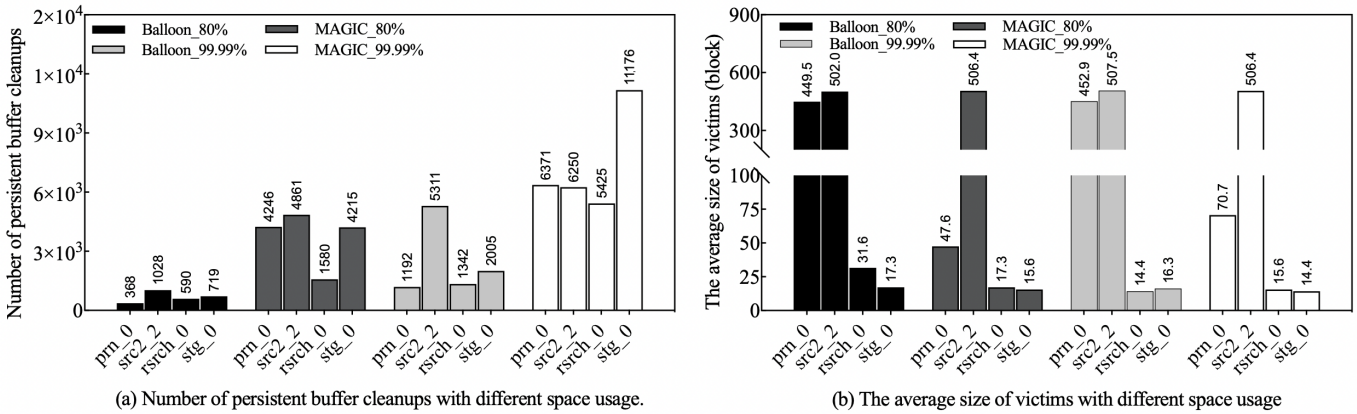


Figure 12. Evaluation of persistent buffer cleanup efficiency of Balloon and MAGIC under different space utilization and workloads.

6. Conclusions

As an emerging magnetic recording technology, IMR can provide higher storage densities and much lower write penalties than SMR. However, with increasing space utilization, RMW operations are frequently triggered, which severely degrades disk performance. In this paper, we investigate how to improve the efficiency of the cooperative work between data placement and data shuffling to mitigate the performance jitter caused by track rewrites. To this end, an elastic data management strategy, the Balloon Transformation Layer, is proposed as a driver-level solution. Specifically, an adaptive write interference data placement policy is proposed to improve data placement efficiency by intelligently placing data in a track with low rewrite probability. Meanwhile, an on-demand data shuffling policy is designed. This policy can not only utilize the buffer tracks in the top region to perform low-cost data migration but also actively place the tracks with high update block coverage in the top region to reduce the write request seek latency. In addition, a write-interference-free persistent buffer is introduced that further reduces track rewrite frequency by elastically adjusting the persistent buffer admission constraint and selectively evicting data blocks. Evaluation results show that Balloon significantly reduces extra I/Os at medium and high space utilization compared with state-of-the-art studies, resulting in an average write performance improvement of up to 68.7% and 78.8%, respectively.

Author Contributions: Conceptualization, C.Z. and F.L.; methodology, C.Z. and W.T.; software, C.Z., F.Y. and M.L.; validation, C.Z., M.L. and W.T.; formal analysis, C.Z., S.L. and W.T.; investigation, C.Z., F.Y., M.L. and S.L.; resources, C.Z., W.W. and F.L.; data curation, C.Z. and W.T.; writing—original draft preparation, C.Z.; writing—review and editing, C.Z., S.L. and W.T.; visualization, C.Z. and W.T.; supervision, S.L., F.L. and W.W.; project administration, W.W., S.L. and F.L.; funding acquisition, S.L., W.W. and F.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 61972311) and partially by ByteDance Inc. (No. PJ20220608900030), Key Basic Research Projects of the Foundation Plan of China (No. 2020-JCJQ-ZD-087) and Shandong Provincial Natural Science Foundation (No. ZR2021LZH009).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liang, Y.; Yang, T.Y.; Yang, M.C. KVIMR: Key-Value Store Aware Data Management Middleware for Interlaced Magnetic Recording Based Hard Disk Drive. In Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC 21), Santa Clara, CA, USA, 14–16 July 2021; pp. 657–671.
2. Li, Y.; Chen, X.; Zheng, N.; Hao, J.; Zhang, T. An exploratory study on software-defined data center hard disk drives. *ACM Trans. Storage* **2019**, *15*, 1–22. [CrossRef]
3. Zhou, S.; Xu, E.; Wu, H.; Du, Y.; Cui, J.; Fu, W.; Liu, C.; Wang, Y.; Wang, W.; Sun, S.; et al. SMRSTORE: A storage engine for cloud object storage on HM-SMR drives. In Proceedings of the 21st USENIX Conference on File and Storage Technologies, Santa Clara, CA, USA, 21–23 February 2023; pp. 395–408.
4. Richter, H.; Dobin, A.; Heinonen, O.; Gao, K.; veerdonk, R.; Lynch, R.; Xue, J.; Weller, D.; Asselin, P.; Erden, M.; et al. Recording on bit-patterned media at densities of 1Tb/in^2 and beyound. *IEEE Trans. Magn.* **2006**, *42*, 2255–2260. [CrossRef]
5. Zhu, J.G.; Wang, Y. Microwave Assisted Magnetic Recording Utilizing Perpendicular Spin Torque Oscillator with Switchable Perpendicular Electrodes. *IEEE Trans. Magn.* **2010**, *46*, 751–757. [CrossRef]
6. Hwang, E.; Park, J.; Rauschmayer, R.; Wilson, B. Interlaced Magnetic Recording. *IEEE Trans. Magn.* **2017**, *53*, 3101407. [CrossRef]
7. Kryder, M.H.; Gage, E.C.; McDaniel, T.W.; Challener, W.A.; Rottmayer, R.E.; Ju, G.; Hsia, Y.T.; Erden, M.F. Heat Assisted Magnetic Recording. *Proc. IEEE* **2008**, *96*, 1810–1835. [CrossRef]
8. Granz, S.; Jury, J.; Rea, C.; Ju, G.; Thiele, J.U.; Rausch, T.; Gage, E.C. Areal Density Comparison Between Conventional, Shingled, and Interlaced Heat-Assisted Magnetic Recording With Multiple Sensor Magnetic Recording. *IEEE Trans. Magn.* **2019**, *55*, 3100203. [CrossRef]
9. Chen, S.H.; Huang, K.H. Leveraging Journaling File System for Prompt Secure Deletion on Interlaced Recording Drives. *IEEE Trans. Emerg. Top. Comput.* **2022**, 1–14. Available online: https://ieeexplore.ieee.org/abstract/document/9976947 (accessed on 1 August 2023). [CrossRef]
10. Liang, Y.; Yang, M.C.; Chen, S.H. MAGIC: Making IMR-based HDD perform like CMR-based HDD. *IEEE Trans. Comput.* **2021**, *71*, 643–657. [CrossRef]
11. Liang, Y.; Yang, M.C. Move-On-Modify: An Efficient yet Crash-Consistent Update Strategy for Interlaced Magnetic Recording. In Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 97–102.
12. Gao, K.; Zhu, W.; Gage, E. Write Management for Interlaced Magnetic Recording Devices. U.S. Patent 9508362, 29 November 2016.
13. Gao, K.; Zhu, W.; Gage, E. Interlaced Magnetic Recording. U.S. Patent 9728206, 8 August 2017.
14. Wu, F.; Li, B.; Zhang, B.; Cao, Z.; Diehl, J.; Wen, H.; Du, D.H.C. TrackLace: Data Management for Interlaced Magnetic Recording. *IEEE Trans. Comput.* **2021**, *70*, 347–358. [CrossRef]
15. Hajkazemi, M.H.; Kulkarni, A.N.; Desnoyers, P.; Feldman, T.R. Track-based Translation Layers for Interlaced Magnetic Recording. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC 19), Renton, WA, USA, 10–12 July 2019; pp. 821–832.
16. Wu, F.; Zhang, B.; Cao, Z.; Wen, H.; Li, B.; Diehl, J.; Wang, G.; Du, D.H.C. Data Management Design for Interlaced Magnetic Recording. In Proceedings of the 10th USENIX Conference on Hot Topics in Storage and File Systems, Boston, MA, USA, 9–10 July 2018; pp. 1–6.
17. Narayanan, D.; Donnelly, A.; Rowstron, A. Write offloading: Practical power management for enterprise storage. *ACM Trans. Storage* **2008**, *4*, 1–23. [CrossRef]
18. SNIA IOTTA Repository: MSR Cambridge Block I/O Traces. Available online: http://iotta.cs.hmc.edu/traces/388 (accessed on 1 August 2023).
19. Park, D.; Du, D.H.C. Hot data identification for flash-based storage systems using multiple Bloom filters. In Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST), Denver, CO, USA, 23–27 May 2011; pp. 1–11.
20. Park, D.; He, W.; Du, D.H.C. Hot Data Identification with Multiple Bloom Filters: Block-Level Decision vs. I/O Request-Level Decision. *J. Comput. Sci. Technol.* **2018**, *33*, 79–97. [CrossRef]
21. Wu, F.; Fan, Z.; Yang, M.-C.; Zhang, B.; Ge, X.; Du, D.H.C. Performance evaluation of host aware shingled magnetic recording (HA-SMR) drives. *IEEE Trans. Comput.* **2017**, *66*, 1932–1945. [CrossRef]

22. Aghayev, A.; Shafaei, M.; Desnoyers, P. Skylight-a window on shingled disk operation. In Proceedings of the 13th USENIX Conference on File and Storage Technologies, Santa Clara, CA, USA, 16–19 February 2015; pp. 135–149.
23. Desktop HDD Product Manual Standard Models st1000dm003 st500dm002. Available online: https://www.seagate.com/www-content/product-content/desktop-hdd-fam/en-us/docs/100768625b.pdf (accessed on 1 August 2023).
24. Zeng, Z.; Chen, X.; Yang, L.; Cui, J. IMRSim: A Disk Simulator for Interlaced Magnetic Recording Technology. In Proceedings of 19th IFIP International Conference on Network and Parallel Computing, Jinan, China, 24–25 September 2022; pp. 267–273.
25. Linux Kernel Libaio: Linux Kernel AIO Access Library. Available online: https://github.com/littledan/linux-aio (accessed on 1 August 2023).
26. Xiao, W.; Dong, H.; Ma, L.; Liu, Z.; Zhang, Q. HS-BAS: A hybrid storage system based on band awareness of shingled write disk. In Proceedings of IEEE 34th International Conference on Computer Design, Phoenix, AZ, USA, 3–5 October 2016; pp. 64–71.
27. Liu, W.; Zeng, L.; Feng, D.; Kent, K.B. ROCO: Using a solid state drive cache to improve the performance of a host-aware shingled magnetic recording drive. *J. Comput. Sci. Technol.* **2019**, *34*, 61–76. [CrossRef]
28. Liang, Y.-P.; Chen, S.-H.; Chang, Y.-H.; Lin, Y.-C.; Wei, H.-W.; Shih, W.-K. Mitigating write amplification issue of SMR drives via the design of sequential-write-constrained cache. *J. Syst. Archit.* **2019**, *99*, 101634. [CrossRef]
29. Ma, C.; Shen, Z.; Wang, J.; Wang, Y.; Chen, R.; Guan, Y.; Shao, Z. Tiler: An autonomous region-based scheme for smr storage. *IEEE Trans. Comput.* **2020**, *70*, 291–304. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.