SLA: A Cache Algorithm for SSD-SMR Storage System with Minimum RMWs

Xuda Zheng[1], Chi Zhang[1], Keqiang Duan[1], Weiguo Wu[1](✉), and Jie Yan[2]

[1] Xi'an Jiaotong University, Xi'an, China
wgwu@mail.xjtu.edu.cn
[2] Hikvision (Beijing) Co., Ltd., Beijing, China

Abstract. To satisfy the low-cost and massive data storage requirements imposed by data growth, Shingled Magnetic Recording (SMR) disks are extensively employed in the area of high-density storage. The primary drawback of SMR disks is the write amplification issue caused by sequential write constraints, which is becoming more prominent in the field of non-cold archives. Although a hybrid storage system comprised of Solid State Disks (SSDs) and SMRs may alleviate the aforementioned issue, current SSD cache replacement algorithms are still limited to the management method of Least Recently Used (LRU). The LRU queue, on the other hand, is ineffective in reducing the triggers of Read-Modify-Write (RMW), which is a critical factor for the performance of SMR disks. In this paper, we propose a new SMR Locality-Aware (SLA) algorithm based on a band-based management method. SLA adopts the Dual Locality Compare (DLC) strategy to solve the hit rate reduction problem caused by the traditional band-based management method, as well as the Relatively Clean Band First (RCBF) strategy to further minimize the number of RMW operations. Experiments indicate that, compared to the MSOT method, the SLA algorithm can maintain a similar hit rate as the LRU, while reducing the number of RMWs by 77.2% and the SMR disk write time by 95.1%.

Keywords: Shingled Magnetic Recording · Replacement algorithm · Hybrid storage system · Spatial locality

1 Introduction

Shingled Magnetic Recording (SMR) increases disk capacity with minimal manufacturing changes. SMR disks remove gaps between tracks by writing tracks in an overlapping way to produce a higher areal density than disks utilizing Conventional Magnetic Recording (CMR) technology [12]. Due to its large capacity and low cost [5], SMR disks are widely utilized in data centers for cold archive

This work is supported by the National Key Research and Development Program (2017YFB1001701), the National Science Foundation of China (61972311), Shandong Provincial Natural Science Foundation (ZR2019LZH007).

storage, and low-frequency storage. However, as the requirement for low-cost storage grows, the issue of how to convert SMR disks to standard storage has become increasingly pressing.

The three types of SMR interface implementations are Host Managed, Drive Managed, and Host Aware, which correspond to the three types of disk devices, HM-SMR, DM-SMR, and HA-SMR, respectively. The DM-SMR disk has a Persistent Buffer (PB) that caches non-sequential write requests and manages sequential write constraints internally, enabling both sequential and random write operations. The Shingled Translation Layer (STL) cleans up the PB periodically [13], which blocks the IO requests and causes a great jitter in application performance. In this paper, DM-SMR disks are used in our algorithms and experiments, and we shall refer to DM-SMR disk as SMR disk in the following section.

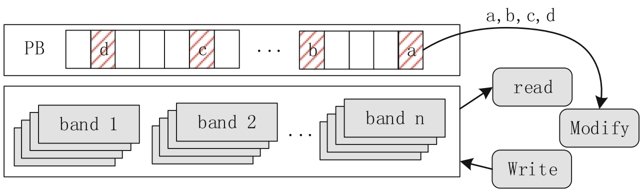


Fig. 1. The process of RMW operation

The SMR disk can only conduct sequential writes, random write data must be stored in the PB to enable in-place update write operations. The Read-Modify-Write (RMW) operation is introduced to write back the buffered data to the band region, When the PB is almost full. As illustrated in Fig. 1, the RMW operation first loads the whole band to which the evicted block belongs into the memory, then all blocks belonging to the band in the PB will be modified, and finally, the entire band will be written back into the SMR band area. While RMW operation solves the sequential write constraints, it also causes significant write amplification and performance fluctuations [4].

A hybrid storage system comprised of SSD and SMR disks relieves the aforementioned issues. However, traditional cache replacement algorithms, such as LRU, do not take the intrinsic characteristics of SMR disks into consideration, resulting in significant write amplification. The trade-off between write amplification and hit rate is the subject of several later cache replacement algorithms for SMR drives [4,6]. Sun et al. [1], on the other hand, utilize Pearson Correlation Coefficients (PCC) to quantify the relativity between the factors and I/O time and demonstrate that the number of RMWs is the key factor affecting the performance of SMR disks. This paper is also based on the above observations.

In this paper, we present the **S**MR **L**ocality-**A**ware (SLA) cache replacement algorithm, which is designed specifically for SSD-SMR hybrid storage system and focuses on how to minimize the number of RMWs. The cache block described in the following refers to the SSD cache block since the algorithm is SSD-oriented. Compared to the MOST algorithm, the SLA algorithm can reduce the number of RMWs by 77.2% and the disk write time by 95.12% while maintaining a hit rate similar to that of the LRU. The main contributions of this paper are as follows.

1) The spatial locality of evicted blocks and the eviction sequence of clean/dirty blocks are proposed as two key factors that influence the frequency of RMWs. Following that, we conducted theoretical analysis and comparison experiments to demonstrate the effectiveness of these two factors, as well as to address the side effects brought about by them.
2) It is presented that increasing spatial locality, i.e. employing a band-based management method, is crucial to minimizing RMWs. We investigate the footprint and recency of the band-based management system to figure out why it has such a poor hit rate. In this paper, The term "footprint" refers to the number of cache blocks per unit band, while "recency" refers to the number of recently accessed blocks per unit band. Following that, a band-based management method based on dual locality (spatial and temporal locality) is suggested to achieve a similar hit rate to LRU.
3) In the band-based management method, we estimate the costs of clean and dirty blocks and propose an Relatively Clean Band First (RCBF) eviction mechanism. The RCBF strategy can evict the cache blocks at the lowest cost, decreasing the number of RMWs even further.

The rest of this paper is organized in the following manner. Section 2 describes the related work. Section 3 presents the motivation for our work. The architecture of the SLA algorithm is described in Sect. 4. The SLA algorithm is evaluated in a simulated environment and on a real disk in Sect. 5. Finally, Sect. 6 concludes this paper with a summary of the SLA algorithm.

2 Related Work

SMR Device. Early discussions on SMR disks focused on the working principle [19,24], STL design [26,28], database [5,9,25,27], and file system [3,10]. Through the skylight approach, Aghayev et al. [2] found the main features of two drive-managed SMR disks, including PB size, band size, and so on, after commercial usage of SMR disks. Skylight also serves as a theoretical foundation for further SMR Disk Research. Ma et al. [7] employs a two-level buffer cache architecture, which keeps hot data in the filter buffer and sends cold data to the SMR disk. K-Framed Reclamation (KFR) [23] reduces performance recovery time by breaking down the reclamation process into more fine-grained KFramed reclamation processes. Ma et al. [11] manages the SMR disks by replacing the PB with a built-in NAND flash to achieve faster cleaning.

DM-SMR Based Hybrid Storage. In a hybrid storage system composed of SSDs and DM-SMRs, current research mainly focuses on SSD cache replacement algorithms. The major cache replacement algorithms based on SSD and

DM-SMR hybrid storage systems can be classified into two categories: LRU-like and Band-based. One of the fundamental components of the LRU-like algorithm is the LRU queue. Since the LRU does not consider the inherent characteristics of SMR disks, it is not beneficial to reduce the number of RMWs. The band-based algorithm can gather the eviction cache blocks as much as possible, thereby reducing the number of RMWs at the source. Currently existing LRU-like algorithms include Partially Open Region for Eviction (PORE) [4] and SMR-Aware Co-design (SAC) [1], and Band-based algorithms include MOST [6]. For the MOST policy, When the cache block needs to be evicted, the band with the **most** cache blocks will be selected.

The PORE algorithm organizes the cache queue using LRU, which limits the LBA range of evicted dirty blocks, lowering the rate of write amplification. However, PORE does not distinguish between clean blocks and dirty blocks. The SAC algorithm extends the PORE algorithm by separating clean and dirty blocks into two LRU queues for management, but its fundamental algorithm remains LRU and does not depart from conventional algorithm restrictions. The MOST algorithm tends to minimize write amplification and for the first time presents a band-based management method. All cache blocks belonging to the same band are grouped together, and the MOST policy evicts the band with the most cache blocks. The degree of aggregation of evicted cache blocks, i.e., spatial locality, can be maximized using the band-based management. However, the MOST algorithm ignores the effect of cache hit rate, which is the fundamental purpose of cache.

HA-SMR Based Hybrid Storage. Current research focuses on the recognition and restructuring of I/O streams in a hybrid storage system made of SSDs and HA-SMRs. ZoneTier [15] is a hybrid storage system that employs SSDs and HA-SMRs to effectively manage all non-sequential writes using a zone-based storage tiering and caching co-design technique. Xie et al. [16] takes use of HA-SMR drives' inherent host-awareness to filter both sequential and innocuous non-sequential writes out of SSD and write them straight to HA-SMR disk. Liu et al. [20] aims to improve the performance of a HA-SMR drive by rearranging out-of-order writes belonging to the same zone and using the SSD cache to absorb update writes and small random writes.

3 Motivation

The data cached in PB is written back in the form of RMW due to the overlapping of the internal tracks of the SMR disk. Correspondingly, the performance fluctuations of SMR disk is caused by RMW. As a consequence, to guarantee the efficiency of the cache, the replacement algorithm must be adaptively adjusted according to the specific storage medium characteristics, i.e. RMW. Therefore, the motivation of this paper is how to effectively minimize the number of RMWs.

The reduction of the number of RMWs is the key factor in optimizing SMR disk performance. The degree of spatial locality of evicted blocks and the sequence of evicted clean/dirty blocks are two major factors that influence

RMWs. We think that the better the locality of the evicted block, that is, the more concentrated the distribution, the fewer RMWs will be triggered. Similarly, since clean blocks do not generate write-back operation, it do not trigger RMW. The remainder of this section will demonstrate the effectiveness of these two factors.

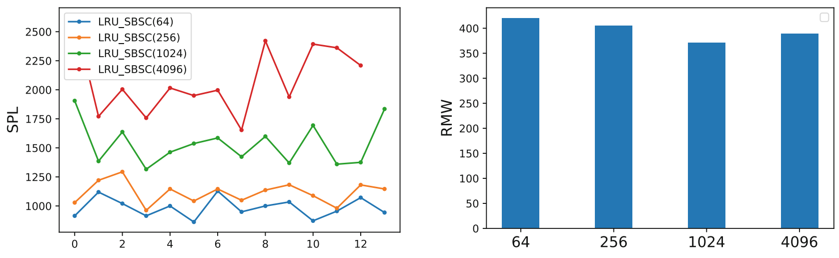
Algorithm 1. LRU_SBSC eviction operation

```
Input: The missed block
Output: The number of blocks to evict
1: if $cache\_full == TREU$ then
2:    $evict\_block = pop\_buf(LRU \rightarrow tail)$
3:    $band\_num = calculate\_band\_number(evict\_block)$
4:    while $cur\_evict\_num < max\_evict\_num$ do
5:       delete_from_LRU(evict_block)
6:       write_to_smr(evict_block)
7:       if $band\_num \rightarrow next \quad != NULL$ then
8:          $evict\_block = band\_num \rightarrow next$
9:       end if
10:   end while
11: end if
12: return $cur\_evict\_num$
```

We design the LRU_SBSC (Same Band, Same Cleaning) algorithm to verify the impact of the spatial locality of evicted blocks. The algorithm organizes the queue in an LRU manner. When the queue is full, the last block in the queue is chosen for eviction, and all blocks in the same band are evicted at the same time. To demonstrate the influence of the evicted block's spatial locality on the performance of the SMR disk, we count the size of the SPL (Spatial Locality) indicator. Equation 1 depicts the $SPL_{average}$ calculation formula. *band_blocks* represents cache blocks for a single band that are evicted in a cycle (a cycle is defined as the process of evicting a certain number of cache blocks), and *band count* represents the number of bands to which these cache blocks belong.

$$SPL_{average} = \frac{\sum_{k=1}^{n} band\_blocks_k}{band\ count} \tag{1}$$

We conducted experiments based on *src* trace(detailed information in Tab. 1) to get realistic $SPL_{average}$ and RMWs values. The abscissa indicates the number of evicted cycles, while the ordinate reflects the $SPL_{average}$ value, as illustrated in Fig. 2(a). The value of $SPL_{average}$ varies dynamically in relation to the granularity of the eviction. The granularity is 64, 256, 1024, and 4096 correspondingly, with each unit representing a 4 KB physical block. The experimental findings are presented in Fig. 2(b), where the number of RMWs reduces as $SPL_{average}$ increases, but increases when $SPL_{average}$ is too high. In Sect. 4.3, we will introduce how to avoid side effects caused by excessive $SPL_{average}$.



(a) Comparison of $SPL_{average}$ under different eviction granulates

(b) The number of RMWs under different $SPL_{average}$

Fig. 2. The effect of $SPL_{average}$ on the number of RMWs

The second factor is the sequence of evicted clean and dirty blocks. The effect of this factor on RMWs is self-evident, since clean blocks do not need to be written back, and no RMW is triggered. However, merely prioritizing the eviction of clean blocks [14] is not a suggested replacement algorithm, since it will have a detrimental impact on the hit rate of the cache. In Sect. 4.4, we will discuss how to circumvent this issue.

4 Design and Algorithm

We developed the SLA algorithm taking into account the substantial effect of the aforementioned two factors on RMW. Section 4.1 primarily introduces three key components of the SLA algorithm. Sections 4.2, 4.3, and 4.4 detail the specific functions of the three components.

4.1 Overview

The architecture of the SLA algorithm is depicted in Fig. 3. The algorithm is made up of three key components: Band-based Management (BM), Dual Locality Compare (DLC), and Relatively Clean Band First (RCBF). We classify cache blocks based on three criteria: physical location, recent access, and clean/dirty data. The data blocks of the same color indicate that they belong to the same band, **F** represents the number of data blocks in the band, **R** represents the data that has been accessed recently, and **C** represents clean data.

The BM component determines its band number by calculating the offset of the cache blocks [1], and it manages and evicts cache blocks belonging to the same band in a consistent manner. The DLC component augments the eviction mechanism with the temporal locality of the cache block and the spatial locality of the band, ensuring the band-based management method's hit rate. To obtain the final eviction sequence, the RCBF component sorts the bands filtered by DLC according to the weight of the clean data block.

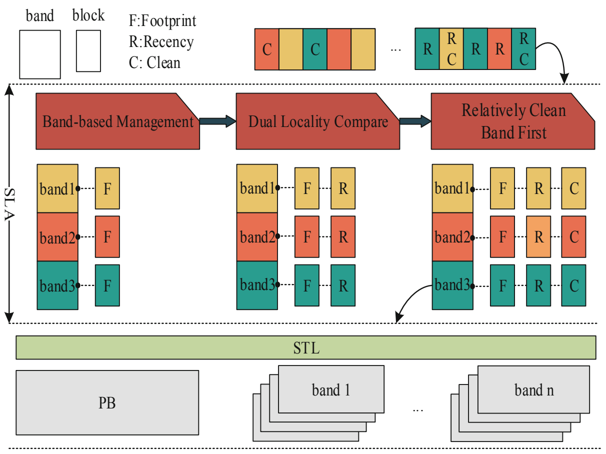


Fig. 3. SLA algorithm architecture

4.2 Band-Based Management

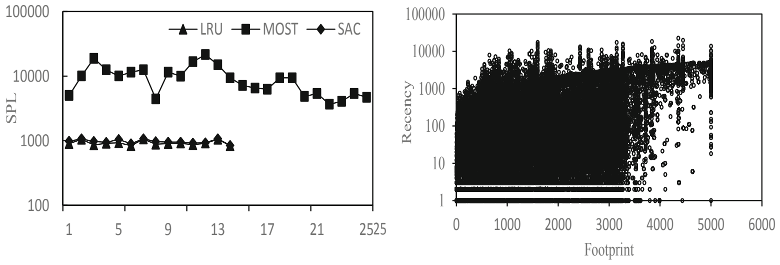
For a long time, the temporal locality of LRU and its variants has been the primary consideration in system cache management, while the spatial locality of cache blocks has been largely ignored [8]. In this paper, we argue that spatial locality is just as important as temporal locality. The reason for this is due to the following two factors. First, compared to the first-level cache, the temporal locality of SSDs is degraded as a second-level cache [22]. Second, SMR disks show unique spatial locality characteristics as a result of the sequential write constraints. Considering the above factors, the SLA algorithm proposed in this paper adopts a more spatially localized band-based management method.

We organize and manage cache blocks in the form of bands, and evict them in units of bands, to enhance the spatial locality of evicted cache blocks. The MOST algorithm has a similar management method, but it is solely dependent on the number of cache blocks in the band, and the band with the most cache blocks is evicted first. This eviction mechanism ensures that the cache block's spatial locality is optimum, but it also significantly lowers the hit rate. Our SLA algorithm employs the Dual Locality Compare and Relatively Clean Band First (RCBF) strategies to minimize RMW while maintaining a high hit rate.

4.3 Dual Locality Compare

Under the band-based management method, all blocks in the cache are statistically sorted according to the bands. When a band is evicted, the band with the most blocks is chosen first. Although this basic method ensures that each evicted block has the greatest spatial locality, it has two apparent flaws. The first disadvantage is that it does not account for the degree of cold and hot, resulting in a relatively low hit rate of the algorithm while lowering overall system performance. The second disadvantage is that the evicted hot block will be re-read into the memory in a short time, reducing the efficiency of the RMW operation.

Before addressing these two flaws, we first explain why the algorithm has such a poor hit rate. We assume that the poor hit rate of band-based management method is related to its locality. We represent temporal locality by recency and spatial locality by footprint here to better quantify the overall locality of the band.



(a) The size of $SPL_{average}$ (b) The distribution of recency and footprint

Fig. 4. Characteristics of traditional band-based algorithms

To obtain an authentic locality, we also conducted experiments based on *src* trace. The footprint and recency of all bands are first counted, and the results are presented in Fig. 4(b). It can be observed that when a band's footprint is high, its recency is comparatively concentrated and high, while when the footprint of a band is low, its recency is more dispersed and low. It should be noted that evicting the band with the most cache blocks first almost always results in the eviction of more hot data blocks. This is why the band-based management method has a poor hit rate.

The $SPL_{average}$ value of the three algorithms, MOST, LRU, and SAC, is then calculated. The number of blocks in a single band is 5000, which must be stated in advance. The reason for this is that DM-SMR drives are made up of bands ranging in size from 15 MB to 40 MB [2], in this paper, we assume the band is 20 MB [1]. The $SPL_{average}$ values of the LRU and SAC algorithms are both about 1000, as shown in Fig. 4(a), while the MOST algorithm is about 10000, which is double the number of blocks per unit band. The $SPL_{average}$ value of MOST algorithm also demonstrates that the same band has been evicted many times in a unit cycle.

$$W_{TPL}\,(b) = \sum F(cur\_time - reference\_time) \tag{2}$$

$$W_{SPL}\left(b\right) = SPL_{band}\left(b\right) \tag{3}$$

Through the above analysis of the spatial and temporal locality of bands, it is observed that the influence of temporal locality and spatial locality on the hit rate of band-based algorithms is comprehensive. To quantify this influencing factor, we express the weight of temporal locality and spatial locality by $W_{TPL}\left(b\right)$ and $W_{SPL}\left(b\right)$, as shown in Eq. 2, 3. The temporal locality calculation Equation of a single data block [17] is $F\left(x\right) = \left(\frac{1}{p}\right)^{\alpha x}$ $(p \geq 2\ ,\ 0 \leq \alpha \leq 1)$, and $SPL_{band}\left(b\right)$ represents the total number of blocks in the entire band. The complexity will grow as we count the access time and computation weight of each block. In this case, we borrow from *Calculus* and consider several blocks accessed at the approximate time as the same computation, or we use the number of recent accesses instead.

$$W_{band}\left(b\right) = \alpha W_{SPL}\left(b\right) - \beta W_{TPL}\left(b\right) + \gamma W \tag{4}$$

The overall weight of the band is expressed as $W_{band}\left(b\right)$. We may deduce from the above explanation that the $W_{band}\left(b\right)$ is inversely proportional to temporal locality and directly proportional to spatial locality. Because the locality value is a linear accumulation, the final calculation formula of $W_{band}\left(b\right)$ is given in Eq. 4. W is the value linked to clean blocks and dirty blocks, as discussed in the next section. The coefficients of these three terms are represented by α, β, and γ, which are all uniformly set to 1.

4.4 Relatively Clean Band First

The cost of clean blocks and dirty blocks in conventional HDDs is not much different. However, because of the write amplification of SMR disks, the cost of evicted dirty blocks is considerably greater than that of clean blocks. Taking this into consideration, we recalculated the eviction costs of clean and dirty blocks, represented by $Cost_{clean}$ and $Cost_{dirty}$, respectively. As shown in the Eq. 5 and Eq. 6, $Cost_{access\_miss}$ and $Cost_{write\_back}$ indicate the time needed to access the missed block and the time required to write the cache block back to disk, respectively. $P\left(access\right)$ represents the probability of the block being re-access.

$$Cost_{clean} = Cost_{access\_miss} * P(access) \tag{5}$$

$$Cost_{dirty} = Cost_{write\_back} + Cost_{access\_miss} * P(access) \tag{6}$$

The eviction mechanism in SLA algorithm takes the band as the unit, so we use $Cost_{band}$ to represent the cost of evicting a band. The calculation formula is as shown in Eq. 7, where N_{clean} and N_{dirty} represent the number of clean and dirty data blocks in the band, and N represents the sum of the two. So the above formula can be simplified to the Eq. 8.

$$Cost_{band} = Cost_{clean} * N_{clean} + Cost_{dirty}\ * N_{dirty} \tag{7}$$

$$Cost_{band} = Cost_{write\_back} * N_{dirty} + Cost_{access\_miss} * N * P(access) \quad (8)$$

The candidate eviction band has a comparable N value in the band-based management method, so it may be considered as a fixed value. Similarly, $P(access)$ can be regarded as a fixed value in a fixed eviction algorithm and trace. As a result, the eviction cost of a band is proportional to the number of dirty blocks evicted, i.e., $Cost_{band} \propto N_{dirty}$.

Given that the number of blocks in the candidate band is about equal, the more clean blocks there are, the fewer dirty blocks there are. As a result, we recommend using the Relatively Clean Band First (RCBF) eviction strategy. The number of clean blocks in the band is taken as a positive factor by RCBF and added to the computation of $W_{band}(b)$, which is the value of W in Eq. 4.

5 Evaluation

This section will conduct a comprehensive evaluation of our proposed SLA algorithm. Section 5.1 introduces the experimental settings. We conduct performance comparison experiments between SLA and other typical caching algorithms in Sects. 5.2 and 5.3 respectively. Section 5.2 verifies the performance of the hit rate, RMWs, and other parameters through the emulator. Then the experiments to evaluate the performance of the SLA algorithm on a real disk are presented in Sect. 5.3.

5.1 Experimental Setup

The experiments are carried out in the Ubuntu 20.04.1 version based on the kernel 5.11.0-34-generic environment, with an AMD EPYC 7302 16-Core CPU @ 3.0 GHz and 16 G DRAM. All experiments are based on the Seagate 8TB SMR drive model ST0008AS0002 [18]. We set the block size to 4 KB and artificially decrease the effective SSD capacity, i.e. the size of the actual cache, to 256 K blocks [4,13]. The experiments use five real-world enterprise I/O traces released by Microsoft Research in Cambridge [21], which represent five different write request ratio traces from 20% to 90%. The specific information is shown in Table 1. Since this algorithm is oriented to standard storage, write-only traces are not used for experiments.

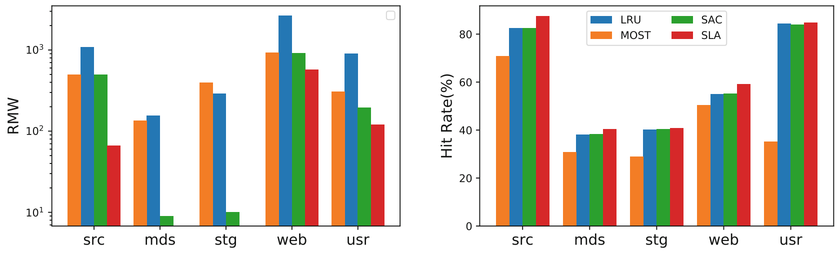
To demonstrate the efficacy of the SLA algorithm, we also compared it to three other algorithms: LRU, MOST, and SAC. LRU and MOST represent the traditional LRU-like algorithm and Band-based algorithm, respectively. SAC stands for the state-of-the-art cache replacement algorithm in the hybrid storage system comprised of SSDs and DM-SMRs. When the SSD cache space is Insufficient, the LRU algorithm chooses the least recently used blocks for eviction, the MOST method chooses the band with the most cache blocks, and the SAC algorithm chooses the least costly blocks for eviction.

Table 1. Real-world traces details

Trace	Total requests (*10^6)	Write percent	Written LBA range (GB)	Accessed LBA range (GB)
src	14	0.832	3.8	3.93
mds	2.9	0.704	3.58	3.73
stg	6	0.682	7.29	7.58
web	9.6	0.464	7.11	8.35
usr	12.8	0.279	6.42	6.92

5.2 Emulation Experiments

Due to the fact that we cannot get the RMW information from the real disk, we performed experiments in a emulator environment on five different traces. SLA achieves a significant advantage compared with its competitors, As illustrated in Fig. 5, we first evaluate the number of RMWs and hit rates produced by various algorithms across five traces. The number of RMWs triggered by the SLA algorithm has reduced to various degrees when compared to LRU-like algorithms such as LRU and SAC, with an average reduction of 91.8% and 72.6%, respectively. When compared to the MOST algorithm, it decreased by 86.6%, 60.7%, and 38.8%, respectively, for *src*,*usr*, and *web*. While *mds* and *stg* do not trigger RMWs, the number of RMWs triggered by the five traces decreased by an average of 77.2%.

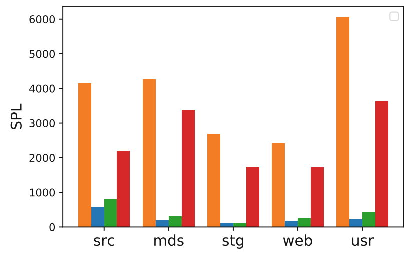


(a) Comparison of the number of RMWs under different traces

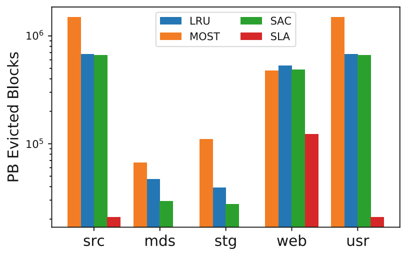
(b) Comparison of hit rates under different traces

Fig. 5. The performance of the four algorithms under different traces

The SLA algorithm maintains the same hit rate as the LRU-like algorithm, which compensates for the poor hit rate of band-based management methods. The hit rate of SLA demonstrates that the reduction in RMW is not achieved by evicting clean blocks at the cost of hit rate, but rather as a consequence of the combination of spatial and temporal locality.



(a) SPL size comparison under different read/write ratios



(b) Comparison of PB evicted blocks under different read/write ratios

Fig. 6. The performance of the four algorithms under different read/write ratios

Further analysis is shown in Fig. 6, the number of PB write-back blocks of SLA algorithm is greatly reduced compared to LRU-like algorithms. Compared to MOST algorithm under src, usr, and web, it is reduced by 98.6%, 92.1%, and 74.2%, respectively. Because there is no RMWs is triggered in mds and stg, the number of PB write-back blocks is 0. In comparison to the MOST algorithm, the $SPL_{average}$ value can be slightly adjusted, demonstrating that the SLA algorithm reduces the probability of the eviction of hot data. Simultaneously, the huge gap in $SPL_{average}$ shows that the spatial locality of band-based management methods, such as SLA and MOST, is far superior to that of LRU-like management methods.

5.3 Real Disk

To verify the performance of SLA and the other three cache replacement algorithms in real SMR disks, we run the above five traces on Seagate ST0008AS0002 8TB SMR. Since the RMW operation only involves the time for the cache blocks to be written to the SMR disk, it has nothing to do with the time for the SSD and reading data from the SMR disk. Therefore, in order to show the performance optimization of SLA more intuitively, we only compare the time it takes for the cache blocks to be written back to the SMR disk.

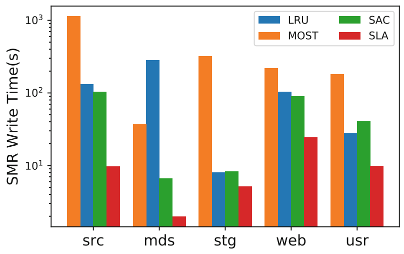


Fig. 7. Disk write time of four algorithms under real SMR disk

The experimental results are shown in Fig. 7. Compared with LRU-like algorithms, the SMR disk write time of the SLA algorithm is greatly reduced. Furthermore, compared with the MOST algorithm, the SLA algorithm reduces under five different traces by 99.1%, 94.8%, 98.4%, 88.8%, and 94.5%, respectively. The disk write time of the five traces decreased by an average of 95.12%.

6 Conclusion

To alleviate the RMWs issue on SMR disks, we propose a new SMR Locality-Aware (SLA) algorithm based on a band-based management method, which optimizes the two key factors affecting RMWs by using DLC and RCBF strategies. Compared with algorithms such as LRU, MOST, and SAC, SLA triggers the least RMWs while maintaining a high hit rate. It is worth mentioning that if the load cannot offer adequate locality for the SLA algorithm, or if the locality is insufficient, the hit rate of SLA will be significantly reduced (the same for LRU-like). To make things worse, since the RCBF strategy is locality-dependent, this issue will be amplified, which is also our next optimization goal.

References

1. Sun, D., Chai, Y.: SAC: a co-design cache algorithm for emerging SMR-based high-density disks. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 1047–1061. ACM, New York (2020)
2. Aghayev, A., Shafaei, M., Desnoyers, P.: Skylight-a window on shingled disk operation. ACM Trans. Storage **11**(4), 1–28 (2015)
3. Chen, S., Yang, M., Chang, Y., Wu, C.: Enabling file-oriented fast secure deletion on shingled magnetic recording drives. In: 2019 56th ACM/IEEE Design Automation Conference, pp. 1–6. IEEE (2019)
4. Wang, C., Wang, D., Chai, Y., Wang, C., Sun, D.: Larger cheaper but faster: SSD-SMR hybrid storage boosted by a new SMR-oriented cache framework. In: Proceedings 33rd International Conference Massive Storage System Technology (2017)
5. Yao, T., Tan, Z., Wan, J., Huang, P., Zhang, Y., et al.: SEALDB: an efficient LSM-tree based KV store on SMR drives with sets and dynamic bands. IEEE Trans. Parallel Distrib. Syst. **30**(11), 2595–2607 (2019)
6. 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: 2016 IEEE 34th International Conference on Computer Design, pp. 64–71. IEEE (2016)
7. Ma, C., Shen, Z., Wang, Y., Shao, Z.: Alleviating hot data write back effect for shingled magnetic recording storage systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **38**(12), 2243–2254(2018)
8. Jiang, S., Ding, X., Chen, F., Tan, E., Zhang, X.: DULO: an effective buffer cache management scheme to exploit both temporal and spatial locality. In: Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies, pp. 1–8. USENIX Association (2005)

9. Liang, Y., Chen, T., Chi, C., Wei, H., Shih, W.: Enabling a B+-tree-based data management scheme for key-value store over SMR-based SSHD. In: 2020 57th ACM/IEEE Design Automation Conference, pp. 1–6. IEEE (2020)
10. Aghayev, A., Ts'o, T., Gibson, G.: Evolving Ext4 for shingled disks. In: 15th USENIX Conference on File and Storage Technologies, pp. 105–120. USENIX Association (2017)
11. Ma, C., Shen, Z., Han, L., et al.: FC: built-in flash cache with fast cleaning for SMR storage systems. J. Syst. Architect. **98**, 214–220 (2019)
12. Gibson, G., Ganger, G.: Principles of operation for shingled disk devices. In: Proceedings 3rd USENIX Workshop Hot Topics in Storage and File Systems, pp. 1–5. USENIX Association (2011)
13. Xie, X., Yang, T., Li, Q., et al.: Duchy: achieving both SSD durability and controllable SMR cleaning overhead in hybrid storage systems. In: Proceedings of the 47th International Conference on Parallel Processing, pp. 1–9. ACM (2018)
14. Park, S., Jung, D., Kang, J., et al.: CFLRU: a replacement algorithm for flash memory. In: Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, pp. 234–241. ACM (2006)
15. Xie, X., Xiao, L., Du, D.H.: ZoneTier: a zone-based storage tiering and caching co-design to integrate SSDS with SMR drives. ACM Trans. Storage **15**(3), 1–25 (2019)
16. Xie, X., Xiao, L., Ge, X., et al.: SMRC: an endurable SSD cache for host-aware shingled magnetic recording drives. IEEE Access **6**, 20916–20928 (2018)
17. Lee, D., Choi, J., Kim, J H., et al.: On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies. In: Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 134–143. ACM (1999)
18. Seagate 2016. http://www.seagate.com/wwwcontent/product-content/hdd-fam/seagate-archive-hdd/enus/docs/100757960h.pdf. Accessed 22 Sept 2021
19. Cassuto, Y., Sanvido, M., et al.: Indirection systems for shingled-recording disk drives. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1–14. IEEE (2010)
20. Liu, W., Zeng, L., Feng, D., et al.: ROCO: using a solid state drive cache to improve the performance of a host-aware shingled magnetic recording drive. J. Comput. Sci. Technol. **34**(1), 61–76 (2019)
21. Narayanan, D., Donnelly, A., Rowstron, A.: Write off-loading: practical power management for enterprise storage. ACM Trans. Storage (TOS) **4**(3), 1–23 (2008)
22. Ye, F., Chen, J., Fang, X., et al.: A regional popularity-aware cache replacement algorithm to improve the performance and lifetime of SSD-based disk cache. In: 2015 IEEE International Conference on Networking, Architecture and Storage, pp. 45–53. IEEE (2015)
23. Ma, C., Wang, Y., Shen, Z., Shao, Z.: KFR: optimal cache management with K-framed reclamation for drive-managed SMR disks. In: 2020 57th ACM/IEEE Design Automation Conference, pp. 1–6. IEEE (2020)
24. Feldman, T., Gibson, G.: Shingled magnetic recording: areal density increase requires new data management. login Mag. USENIX SAGE **38**(3), 22–30 (2013)
25. Yao, T., Wan, J., Huang, P., et al.: GearDB: a GC-free key-value store on HM-SMR drives with gear compaction. In: 17th USENIX Conference on File and Storage Technologies, pp. 159–171. USENIX Association (2019)
26. He, W., Du, D.H.: Novel address mappings for shingled write disks. In: 6th USENIX Workshop on Hot Topics in Storage and File Systems, pp. 1–5. USENIX Association (2014)

27. Chen, S., Liang, Y., Yang, M.: KVSTL: an application support to LSM-tree based key-value store via shingled translation layer data management. IEEE Trans. Comput. (2021)
28. Yang, T., Wu, H., Huang, P., et al.: A shingle-aware persistent cache management scheme for DM-SMR disks. In: 2017 IEEE International Conference on Computer Design, pp. 81–88. IEEE (2017)