MU-RMW: Minimizing Unnecessary RMW Operations in the Embedded Flash with SMR Disk

Chenlin Ma[†], Zhuokai Zhou[†], Yingping Wang[†], Yi Wang[†] and Rui Mao[†‡]
[†]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*
[‡]*Shenzhen Institute of Computing Sciences, Shenzhen, China*
chenlin.ma@szu.edu.cn, {2110276021, 2100271073}@email.szu.edu.cn, {yiwang, mao}@szu.edu.cn

***Abstract*—Emerging Shingled Magnetic Recording (SMR) Disk can improve the storage capacity significantly by overlapping multiple tracks with the shingled direction. However, the shingled-like structure leads to severe write amplification caused by RMW operations inner SMR disks. As the mainstream solid-state storage technology, NAND flash has the advantages of tiny size, cost-effective, high performance, making it suitable and promising to be incorporated into SMR disks to boost the system performance. In this hybrid embedded storage system (i.e., the Embedded Flash with SMR disk (EF-SMR) system), we observe that physical flash blocks can contain a mixture of data associated with different SMR data bands; when garbage collecting such flash blocks, multiple RMW operations are triggered to rewrite the involved SMR bands and the performance is further exacerbated. Therefore, in this paper, we for the first time present MU-RMW to guarantee data from different SMR bands will not be mixed up within the flash blocks with an aim at minimizing unnecessary RMW operations. The effectiveness of MU-RMW was evaluated with realistic and intensive I/O workloads and the results are encouraging.**

***Index Terms*—Shingled Magnetic Recording, NAND Flash, Address mapping, Cache management**

I. Introduction

With the demand for storage capacity increasing drastically in the big data era, Shingled Magnetic Recording (SMR) technique has been proposed as one of the most promising alternatives to break through the areal density limit of Conventional Magnetic Recording (CMR) disks with low cost by overlapping multiple data tracks with the shingled-like structure. However, this shingled-like structure prevents SMR disks from serving random write requests since randomly writing to a specific track will destroy stored data on its adjacent tracks. To conquer this, a time-consuming operation named Read-Modify-Write (RMW) operation is performed by moving the magnetic read/write head back and forth to read out stored data in the SMR disk, and after merging with the newly written data, the involved data are written back to the tracks sequentially, which can raise high responding time and severe write amplification phenomenon.

To alleviate the negative effect brought by RMW operations, a first-level persistent cache (PC) is organized in an appending write mode to avoid random writes to the SMR disk. Several prior works have considered leveraging the advantages of tiny size, cost-effective, and high performance of NAND flash-based storage media (e.g., solid-state-disk (SSD) drive) to boost the overall system performance of SMR disks [1–6]. As random writing to NAND flash-based storage media can also introduce write amplification, an address mapping between the SMR addresses and the flash-based PC should be carefully orchestrated. We argue that the address mapping in the state-of-the-art schemes [2] and [7] are not fully optimized, which lead to a gathering of mixture data associated with different data bands into a single physical flash block. Therefore, when reclaiming such flash blocks, multiple RMW operations are triggered to rewrite all the involved SMR bands, which incur significant performance overhead.

The main idea of this paper is to regulate the writes into each physical flash block so that a single block can only be associated with a particular SMR band. In this way, when reclaiming the flash space, only limited RMW operations will be triggered. To this end, we propose a novel shingled translation layer (STL) called MU-RMW which can mitigate as many unnecessary RMW operations as possible. However, to ensure the feasibility of the proposed MU-RMW scheme, the following challenges must be addressed including: (1) how to redesign the address mapping to better joint-manage both the flash-based PC and SMR bands in the system; (2) how to minimize unnecessary RMW operations so as to improve the overall system performance of the EF-SMR disk; (3) how to perform an efficient cleaning to avoid a time-consuming pause to the host system.

To address the first challenge, a *concentrated address mapping* is utilized to map a group of physical flash blocks to each SMR band on demand so as to avoid a mixture of data from multiple bands. As the SMR disk is of Terabyte-level capacity, maintaining a sector-level mapping table in RAM memory can easily lead to out-of-memory issues. Therefore, the concentrated address mapping breaks the sector-level mapping table into multiple parts and only leaves the band-level related mapping in memory while the remaining parts are stored onto the spare area (i.e., OOB) of each physical flash page.

To handle the second and the third challenges, a *lazy RMW reclamation* strategy is adopted to clean PC according to the system state. Specifically, when reclaiming the flash space during a garbage collection process, MU-RMW will first look for any obsolete flash blocks (i.e., a fully invalidated block). In this case, no RMW operation is needed; otherwise, an SMR band with the maximal mapped flash blocks will be selected as the victim, and thus, the still-valid data will be written to the SMR band with only one single RMW operation.

The main contributions of this paper are summarized as follows:

978-3-9819263-6-1/DATE22/©2022 EDAA

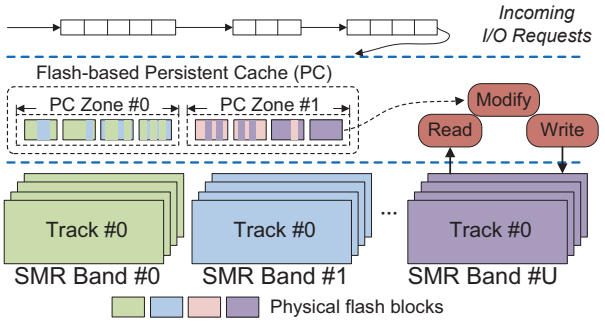

Fig. 1: An illustration of the state-of-the-art scheme named Skylight [7].

- We propose a novel shingled translation layer scheme named MU-RMW in flash-SMR hybrid embedded storage system. A concentrated address mapping approach is presented to co-manage both the flash-based PC and the SMR disk.
- We have adopted a lazy RMW reclamation policy to avoid as many unnecessary RMW operations as possible so as to minimize the overhead of PC cleaning.
- We have built a trace-driven flash-SMR simulator and evaluated our proposed MU-RMW with various realistic and intensive I/O workloads. Compared with the baseline schemes, MU-RMW can reduce the overall system response time by 38.58%, 69.38%, and a maximum reduction of 50.09%, 75.64% respectively.

II. Background and Motivation

A. Flash-based PC in Skylight

SMR disk will devote a small fraction of the SMR space [7–10] (1%∼10%) as a PC to alleviate the RMW effect. The flash-based PC contains multiple PC Zones, and a PC Zone is formed by a certain number of physical flash blocks, as shown in Figure 1. Despite the PC space, the remaining SMR disk space consists of multiple SMR Bands. Note that the capacity of a PC Zone is identical to an SMR Band (e.g., a 40 MB Band size).

As shown in Figure 1, the flash-based PC is served as the first-level cache/buffer for the incoming I/O requests from the host system. Moreover, a flash block (i.e., the basic unit of erase operation) can contain a fixed number of physical flash pages (i.e., the basic read/write unit); a page is composed of a data area and a spare out-of-band (OOB) area [11]. In Skylight, each PC Zone is statically mapped/associated with a certain number of SMR Bands as reflected in the source codes [12].

As long as a PC Zone is full, Skylight will trigger a cleaning process to reclaim the occupied flash blocks by writing back all the buffered valid data within the PC Zone to their associated SMR Bands. Even writing back a single valid data to its associated SMR Band can incur a full SMR Band rewriting performed by the RMW operation including: (1) sequentially read out all the stored data within the SMR Band; (2) merging the valid/modified data; (3) sequentially write back the involved data in a track-by-track manner. Since the SMR disk highly relies on the magnetic read/write head to perform read/write operations, RMW is a time-consuming operation that should be avoided as much as possible.

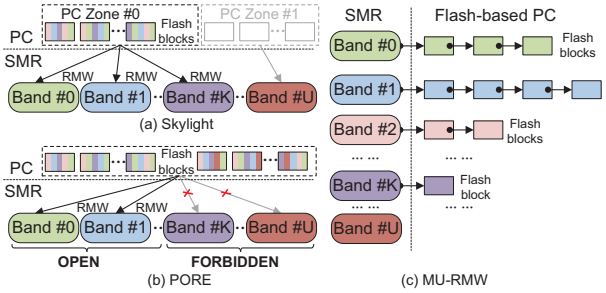

Fig. 2: A comparison between Skylight [7], PORE [2] and our proposed MU-RMW.

B. Motivation

As shown in Figure 2 (a), in Skylight, PC Zone #0 is statically mapped with (K+1) SMR Bands, and data from different bands are appended within each physical flash block. When PC Zone #0 is full, all the valid data are copied from the involved flash blocks and are written back to their associated bands via RMW operations; the involved flash blocks can then be erased and reclaimed for future usage. As shown in Figure 2 (a), in the worst case, PC Zone #0 contains data from all its (K+1) associated SMR Bands, and thus, it will take (K+1) × RMW to reclaim PC Zone #0. Suppose that (K+1) is 128 and an RMW operation takes around 80 ms to finish, then it will take around 10 seconds to clean a single PC Zone, which can significantly degrade the system performance.

In order to promote better performance, the work in [2] proposes a framework named PORE (i.e., Partially Open Region for Eviction) to restrict the LBA range of evicted data from the flash-based PC cache to the SMR disk. All the flash blocks in PC can be associated with any SMR Bands in PORE. As shown in Figure 2 (b), in PORE, SMR Bands are tagged as "OPEN" or "FORBIDDEN" to regulate the evictions during a cleaning process. The basic idea of PORE is that: while data from the "OPEN" region can be evicted by invoking RMW operations, data from the "FORBIDDEN" region should remain in the cache. Since the number of "OPEN" SMR Bands is smaller than (K+1), fewer RMWs are needed within a single cleaning process. Although PORE can alleviate the RMW issues to some extent, we argue that the work is still not fully optimized.

Statically mapping SMR Bands to PC can cause a mixture of data gathering within flash blocks and pose challenges on reclaiming the flash space (i.e., introducing numerous RMWs). Compared to Skylight and PORE, in MU-RMW, we neither group the flash blocks into a PC Zone nor statically associate the Zone with multiple SMR Bands. Instead, we propose to map flash blocks in the PC space dynamically to each SMR Band on demand, as shown in Figure 2 (c). There are several advantages by doing so: first, popular SMR Band can utilize more flash space to serve update requests in an appending mode; second, MU-RMW guarantees each flash block contains data from just one SMR Band; third, when the available PC

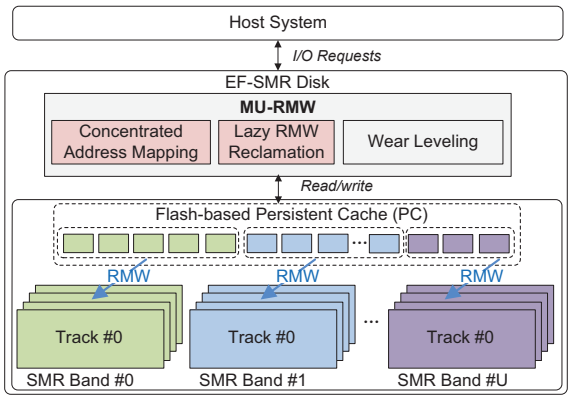

Fig. 3: The system overview of the proposed MU-RMW.

space is in shortage, the SMR Band with the longest linked-list of flash blocks can be chosen and a single RMW operation can reclaim its assigned flash blocks (e.g., Band #1 as shown in Figure 2 (c)).

III. MU-RMW: An Approach to Minimize Unnecessary RMW Operations

A. System Overview

As shown in Figure 3, the proposed MU-RMW management scheme is deployed as a shingled translation layer (i.e., STL) within the controller of the EF-SMR disk. Therefore, MU-RMW can be utilized as a traditional block device that hides the flash-based persistent cache and minimizes the RMW operations by implementing the modules including (1) the concentrated address mapping and (2) the lazy RMW reclamation.

B. Concentrated Address Mapping

A dynamic address mapping is required to ensure data from an SMR Band are concentratively gathered within a set of allocated flash blocks. However, a sector-level mapping table can consume a vast amount of RAM space. For instance, for a terabyte-level SMR disk, maintaining a full sector-level mapping table can consume gigabyte-level RAM space, which is unacceptable and unrealistic within the SMR device. Therefore, in MU-RMW, we propose to adopt a hybrid-level concentrated address mapping (CAM), which can break down the whole sector-level mapping table into several pieces and save them into the OOB spare area for the sake of saving RAM space with negligible performance overhead.

As shown in Figure 4, the logical sector number (LSN) is divided into logical band number (LBN) and band offset (BO) which is the offset within the SMR band. M physical blocks are concentratively allocated to the SMR band on demand and the physical block numbers (BLKs) of these allocated blocks are recorded in the block mapping table (BMT) in a linked-list structure. Specifically, the whole sector mapping table is divided into N sub-tables and these sub-tables are distributed to the OOB area of the flash pages; the sector mapping directory (SMD) resides in RAM space, which consists of N pointers referencing the physical page numbers (PPN) where the sector mapping sub-tables are stored.

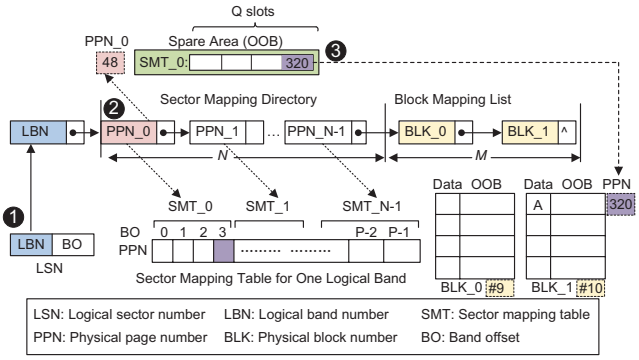

Fig. 4: The proposed concentrated address mapping (CAM) in MU-RMW.

Suppose that an SMR band contains P sectors so that the entire sector mapping table for one logical SMR Band has P entries. We further assume that the OOB area of each physical flash page can hold Q mapping slots of the entire sector mapping table. Therefore, we can derive N=P/Q. In this way, the written data associated with one single SMR band are concentratively gathered within a group of flash blocks and distributed closer to each other.

C. MU-RMW Read and Write Operations

1) MU-RMW Writes: In MU-RMW, all the write requests from the file system will be served in flash-based PC by adopting the concentrative address mapping. A write request consists of the associated data and an LSN, as shown in Figure 5. Dividing the input LSN by P (i.e., the number of sectors in one band), the quotient is the LBN while the remainder is the BO.

By looking up CAM with the derived LBN as the index, we can obtain the block mapping list and pointers of sector mapping subtables of this band. When the data is to be written to a new page, the latest sector mapping subtable will be retrieved first from the OOB area of the PPN referenced by its pointer. After updating the PPN located by PPN_INDEX, the subtable will be written to the allocated new page along with the data and then the pointer of this subtable will be updated accordingly.

Figure 5 serves as an illustrative example of how write operations are performed in MU-RMW. For the sake of illustration, we assume that each band has P=10240 sectors, each block has 32 pages and the available mapping slots of each sector mapping subtable are Q=64. Based on the setting, N equals to 10240/64=160 which indicates that the sector mapping table is divided into 160 subtables (i.e., *PPN_0* to *PPN_159*). Initially, both pointers and the block mapping list are empty.

For the third write request *write(C,24614)*, the *LBN, BO* is (2,4134), respectively. Dividing BO by Q=64, the quotient is the PPN_INDEX (i.e., *PPN_64*) while the remainder is the offset inner the SMT_sub-table (i.e., SMT_OFF=38). A next free page *PPN=350* in BLK #10 is allocated to serve the write request and before data C can be written, the previous version of SMT_sub-table should be retrieved first from *PPN_64*=349. Then SMT_sub-table[38] is updated to reference *PPN=350* and

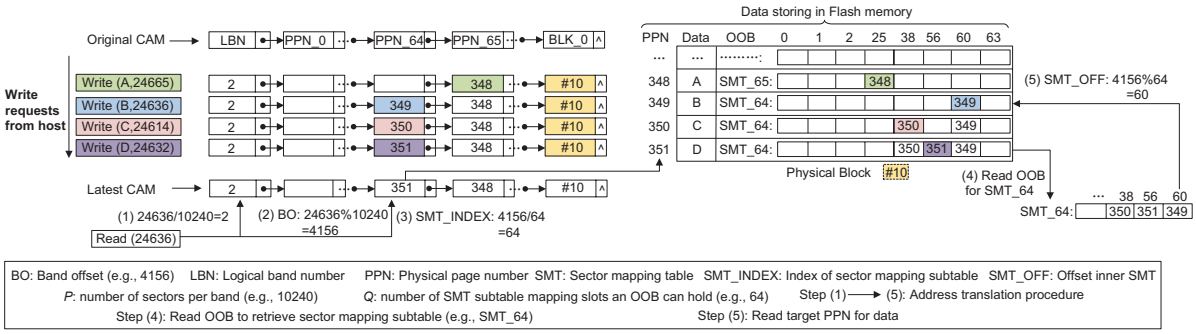

Fig. 5: An illustrative example of read and write operations in MU-RMW.

is written along with *C* to the OOB area and data area. Finally, *PPN_64* within CAM should be updated to reference the PPN where the latest SMT_sub-table is stored. Similarly, other write requests are served.

2) MU-RMW Reads: Unlike a write request, a read request is only represented in a "read(LSN)" form. The pointer of the corresponding mapping subtable will be located according to the corresponding LBN and BO. An illustrative example of the address translation is shown in Figure 5. Specifically, (1) - (2) obtain the pointer *PPN_64=351* according to *LBN=2* and *BO=4156*; (3) read out the corresponding subtable in the OOB area of *PPN=351*; (4) locate the data page *PPN=349* by *SMT_OFF=60* based on the calculation; (5) read the valid data *B* in *PPN=349*.

D. Lazy RMW Reclamation

In MU-RMW, the number of valid pages copy and block erase counts are nontrivial performance overheads that should be considered and minimized. By lazily delaying the reclamation process, a valid page may become invalid so that the number of invalid pages can be increased (resp. the number of valid pages copy can be reduced). As CAM is adopted in MU-RMW, flash blocks are mapped to the same logical SMR Band in an on-demand fashion and the lazy RMW reclamation process will not be invoked until all of the free blocks are consumed. Specifically, as shown in Algorithm 1, the reclamation process follows the steps in the following:

Algorithm 1: Lazy RMW Reclamation

```
Input: Concentrated Address Mapping
Output: Reclamation completed
 1: for LBN = 0 to U do
 2:     while BLK = Iterate(CAM[LBN]→BML) do
 3:         if Is_Obsolete(BLK) then
 4:             Erase_Block(BLK)
 5:             return
 6:         end if
 7:     end while
 8:     if Length(CAM[LBN]→BML) > Longest_Length then
 9:         Longest_Length = Length(CAM[LBN]→BML)
10:         Longest_LBN = LBN
11:     end if
12: end for
13: SMT = Reconstruct_SMT(CAM[Longest_LBN]→SMD)
14: Valid_Data = Read_Valid(SMT)
15: Do_RMW(native(Longest_LBN), Valid_Data)
16: Erase_Blocks(CAM[Longest_LBN]→BML)
```

1. *Select the victim:* An obsolete block (i.e., block with no free page and no valid page) is the ideal victim block in MU-RMW as it will not invoke RMW operation in reclaiming process. To this end, such blocks will be selected and reclaimed in the first place (Line 2-7). Otherwise, the SMR Band with the longest block mapping list is selected as a victim since it has greater chances to have the least valid pages (Line 8-11). All the blocks allocated to this band are then selected as victim blocks.

2. *Migrate valid data and Erase blocks:* After selecting the victim, the sector mapping directory (i.e., SMD) is utilized to reconstruct the entire sector mapping table (i.e., SMT) (Line 13). By looking up SMT, the corresponding PPNs where valid data are stored can be retrieved; then, valid data can be read out by performing flash page read operations to these PPNs (Line 14). At this point, these valid data can be migrated to the associated SMR Band by just a single RMW operation (Line 15), while in the meantime, the involved flash blocks can be erased in parallel (Line 16).

Unlike reclaiming a single flash physical block via numerous RMW operations in Skylight, MU-RMW can reclaim much more free space (i.e., a linked-list of blocks) through only one RMW operation by adopting the lazy RMW reclamation policy. Therefore, the overall performance, such as the average response time, can benefit from such proposed policy.

E. Memory Overhead Analysis

Both Skylight and PORE adopt a sector-level mapping table (i.e., SMT) to record the mapping between each logical sector number and the physical page number, and thus, it can consume a significant amount of memory storage overhead. As mentioned above, P logical sectors form a logical band, and there are U logical band numbers (i.e., LBN), 4 bytes are required to represent one specific logical sector number. Therefore, the total RAM space needed is $P \times U \times 4$ bytes. Let P and U be 10240 and 256, respectively, and thus, the memory overhead of SMT in Skylight and PORE is about 10 MB.

In terms of MU-RMW, each LBN in the CAM is mapped with an SMD and a BML. Each item in SMD requires 4 bytes and there is N number of items in SMD; thus, SMD consumes $4 \times N$ bytes. Therefore, for all U LBNs, the memory space overhead for mapped SMDs is $U \times (4 \times N)$ bytes. Suppose

TABLE I: The characteristics of the traces.

Traces	Data Written (GB)	Data Read (GB)	Write Ratio	Update Ratio
rsrch_0	4.96	0.51	90.68%	99.92%
src1_2	5.43	1.85	74.63%	99.87%
src2_0	5.27	0.67	88.66%	99.86%
src2_2	3.07	1.34	69.67%	99.84%
stg_0	6.57	1.18	84.81%	99.92%
stg_1	3.04	5.34	36.25%	99.92%
ts_0	5.66	1.21	82.42%	99.86%
usr_0	5.09	3.45	59.58%	99.88%
wdev_0	3.49	0.88	79.92%	99.76%
web_0	5.43	2.31	70.12%	99.82%

that there are S physical flash blocks and each pair of <block, pointer> in BML consumes 8 bytes (4 bytes for a block and 4 bytes for a pointer); thus, the maximum size of BML will be $S \times 8$ bytes. Let N and S be 160 and 640, respectively; the total memory space needed for CAM is about 69 KB. Our scheme shows a significant reduction in RAM cost compared with Skylight and PORE.

IV. Evaluation

A. Experimental Setup

We build a trace-driven simulator to emulate the EF-SMR disk by heavily revising the source codes of Skylight [12]. In terms of a fair comparison, the parameters used in the configuration are referred as those in Skylight [7, 13] and PORE [2]. Therefore, in our EF-SMR system, the disk has a rotation speed of 7200 rpm; the write-head width is three times as large as the read-head width which indicates that writing to a certain track will destroy stored data on the adjacent two tracks. The Zone/Band size is configured as 40 MiB, and the flash-based PC consists of two PC Zones in Skylight; the total number of SMR Bands is 256.

In flash-based PC, the latency for one page read, one page write and one block erase are $0.2ms$, $1.3ms$, and $1.5ms$, respectively [14].

Skylight, PORE, and the proposed MU-RMW scheme are evaluated and compared by replaying real-world traces on the emulated EF-SMR disk. The traces are downloaded from data centers of Microsoft Research Cambridge [15]. The detailed characteristics of the traces, including data written (GB), data read (GB), write ratio (%), and update ratio (%), are shown in Table I. Specifically, the write ratio is calculated as the total number of write requests over the total number of read and write requests. The update ratio is calculated as the total number of update requests over the total number of write requests. Among these ten traces, five traces are write-intensive

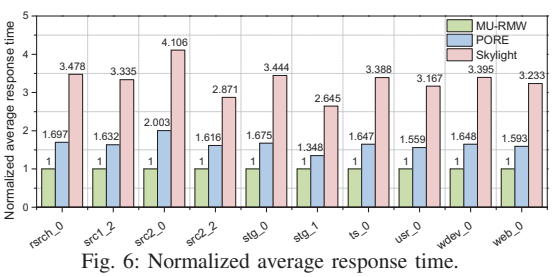

Fig. 6: Normalized average response time.

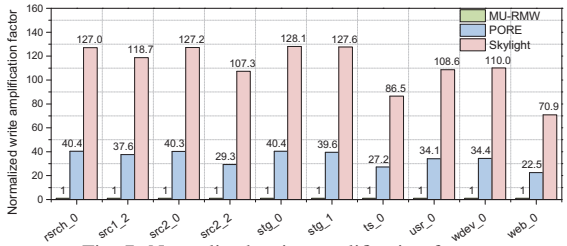

Fig. 7: Normalized write amplification factor.

workloads, four traces are write-medium workloads, and the remaining one trace is read-intensive workload. Each trace is named in a *hostname_disknumber* form. For example, the *stg_1* trace refers to web staging and the disk number is 1.

B. Performance Evaluation

In this section, we compare the following performance metrics for MU-RMW, PORE [2], and Skylight [7]: (1) the average response time; (2) the write amplification factor; (3) the average cleaning time; (4) the worst case response time.

1) Average response time: As the ultimate goal of our MU-RMW scheme is to improve the overall system performance, we first measure the average response time of Skylight, PORE, and MU-RMW in order to perform a quantitative analysis. The average response time is calculated as the total time spent on finishing all the requests over the total number of requests. Among all the traces, we normalize the average response time of PORE and Skylight to that of MU-RMW, and the experimental results are shown in Figure 6. It can be observed that on average, MU-RMW can shorten the overall average response time by 38.58%, 69.38%, and a maximum reduction of 50.09% (i.e., among trace src2_0), 75.64% (i.e., among trace rsrch_0) compared to PORE and Skylight, respectively. The reason why the performance of MU-RMW is superior to both PORE and Skylight is that most RMW operations are avoided, as RMW is the dominating performance overhead in the EF-SMR system. The results prove the effectiveness of the proposed concentrated address mapping and the lazy RMW reclamation policy.

2) Write amplification factor: Since the SMR Band rewriting during an RMW operation can lead to a large write amplification factor (i.e., WAF is calculated as the total written data over the initially written data), the larger WAF, the worse write performance. Therefore, to evaluate the effectiveness of MU-RMW in reducing WAF, we then measure the WAF of Skylight, PORE, and MU-RMW among all the traces. We normalized the WAF of PORE and Skylight to that of MU-RMW, and the results are shown in Figure 7. As shown in Figure 7, the reductions of WAF in MU-RMW range from 22.48X (resp. 70.92X) to 40.4X (resp. 128.09X) comparing with PORE (resp. Skylight). The results show that the SMR Band rewriting in MU-RMW is significantly reduced, and thus, leading to a minimized WAF as well as RMW operations.

3) Average cleaning time: Average cleaning time (ACT) can be utilized to evaluate the overall efficiency of the PC cleaning process to reclaim a certain amount of PC space for future usage. ACT is calculated as the total time spent on the cleaning

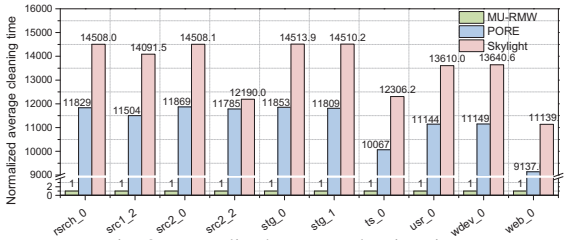

Fig. 8: Normalized average cleaning time.

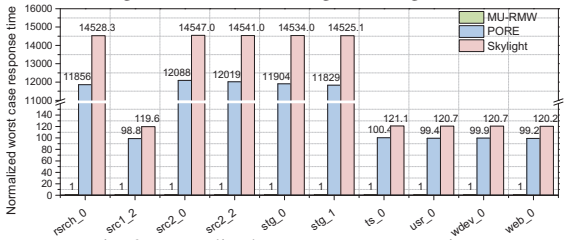

Fig. 9: Normalized worst case response time.

processes over the total cleaning counts. As shown in Figure 8, we normalized the ACT of PORE and Skylight to that of MU-RMW and collected the results.

The ACTs of PORE and Skylight are in second-level while those of MU-RMW are in millisecond-level, and the vast reduction of ACT mainly benefits from adopting the lazy RMW reclamation policy. Specifically, in each cleaning process, compared with Skylight and PORE that have up to tens of hundreds of RMW operations triggered; no RMW or just one RMW is needed in MU-RMW. Therefore, not only the rewritten data is largely reduced, but the overall cleaning time is significantly shortened.

4) Worst case response time: The time-consuming reclamation process can incur runtime performance delay and lead to system pause (i.e., blocking user requests) and thus, can severely degrade the user experience and the quality of service. As the worst case response time (WCRT) is typically utilized to measure the most significant system pause, we collected the WCRT of MU-RMW, PORE, and Skylight among all the evaluated traces for comparison.

As shown in Figure 9, compared with PORE and Skylight, the WCRTs of MU-RMW have several orders of magnitude reduction among rsrch_0, src2_0, src2_2, stg_0, and stg_1, since RMW operation is not even triggered among these traces. Even in the cases that RMW operations are triggered (i.e., src1_2, ts_0, usr_0, wdev_0, and web_0), MU-RMW can still shorten 99.57X and 120.47X WCRT on average compared to PORE and Skylight, respectively. The reasons behind these performance gains include: (1) the reclamation scale is much smaller (e.g., block-level in MU-RMW versus Zone-level in Skylight and PC-level in PORE); (2) unnecessary RMWs and SMR Band rewriting are greatly minimized. The results prove that the proposed MU-RMW effectively minimizes WCRT so as to improve the quality of user experience.

V. Conclusion and Future work

In this paper, we have studied the problem of minimizing unnecessary RMW operations in the embedded flash with SMR (i.e., EF-SMR) disk storage system. Specially, we have proposed a novel STL named MU-RMW with the EF-SMR disk. By adopting the proposed concentrated address mapping and lazy RMW reclamation, our MU-RMW can effectively reduce the number of unnecessary RMW operations. Our experimental results show that MU-RMW can achieve more encouraging performance when compared with previous works.

Several possible directions can be explored in the future. First, we plan to add an independent wear-leveling module to handle the flash blocks allocation issue. Second, in the future work, we plan to reserve an index cache to cache some hot sector mapping subtables to avoid accessing OOBs each time a specific data is required.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (62072311, 61972259, 62122056, 62102263, and U2001212), in part by the Guangdong Basic and Applied Basic Research Foundation (2020B1515120028 and 2019B151502055), and in part by the Shenzhen Science and Technology Program (JCYJ20210324094402008 and JCYJ20210324094208024). Yi Wang is the corresponding author.

References

[1] W.-G. Liu, L.-F. Zeng, D. Feng, and K. B. Kent, "ROCO: Using a solid state drive cache to improve the performance of a host-aware shingled magnetic recording drive," *Journal of Computer Science and Technology*, pp. 61–76, 2019.

[2] C. Wang, D. Wang, Y. Chai, C. Wang, and D. Sun, "Larger cheaper but faster: SSD-SMR hybrid storage boosted by a new SMR-oriented cache framework," in *MSST*, 2017.

[3] D. Sun and Y. Chai, "SAC: A Co-Design Cache Algorithm for Emerging SMR-based High-Density Disks," in *ASPLOS*, 2020, pp. 1047–1061.

[4] W. Xiao, H. Dong, L. Ma, Z. Liu, and Q. Zhang, "HS-BAS: A hybrid storage system based on band awareness of Shingled Write Disk," in *ICCD*, 2016, pp. 64–71.

[5] X. Xie, T. Yang, Q. Li, D. Wei, and L. Xiao, "Duchy: Achieving both SSD durability and controllable SMR cleaning overhead in hybrid storage systems," in *ICPP*, 2018, pp. 1–9.

[6] C. Ma, Z. Shen, L. Han, R. Chen, and Z. Shao, "FC: Built-in flash cache with fast cleaning for SMR storage systems," *Journal of Systems Architecture (JSA)*, pp. 214–220, 2019.

[7] A. Aghayev, M. Shafaei, and P. Desnoyers, "Skylight- Window on Shingled Disk Operation," in *FAST*, 2015, pp. 1–16.

[8] Y. Cassuto, M. A. Sanvido, C. Guyot, D. R. Hall, and Z. Z. Bandic, "Indirection systems for shingled-recording disk drives," in *MSST*, 2010, pp. 1–14.

[9] M.-C. Yang, Y.-H. Chang, F. Wu, T.-W. Kuo, and D. H. Du, "On improving the write responsiveness for host-aware SMR drives," *Transactions on Computers (TC)*, pp. 111–124, 2018.

[10] ——, "Virtual persistent cache: Remedy the long latency behavior of host-aware shingled magnetic recording drives," in *ICCAD*, 2017, pp. 17–24.

[11] R. Chen, Q. Guan, C. Ma, and Z. Feng, "Delay-based I/O request scheduling in SSDs," *Journal of Systems Architecture (JSA)*, pp. 434–442, 2019.

[12] "Drive-Managed SMR Performance Model," http://sssl.ccs.neu.edu/skylight.

[13] F. Wu, B. Li, Z. Cao, B. Zhang, M.-H. Yang, H. Wen, and D. H. Du, "Zonealloy: Elastic data and space management for hybrid SMR drives," in *HotStorage*, 2019, pp. 1–6.

[14] Samsung-Corporation, "K9G4G08U0F 2GB MLC NAND Flash datasheet," https://datasheetspdf.com/pdf/776555/Samsung/K9GAG08U0F/6.

[15] "MSR Cambridge Block I/O Traces," http://iotta.cs.hmc.edu/traces/388.