Le Q, Amer A, Holliday J. RAID 4SMR: RAID array with shingled magnetic recording disk for mass storage systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(4): 854–868 July 2019. DOI 10.1007/s11390-019-1946-4

RAID 4SMR: RAID Array with Shingled Magnetic Recording Disk for Mass Storage Systems

Quoc Le, Ahmed Amer, and JoAnne Holliday, *Member*, *ACM*, *IEEE*

Department of Computer Engineering, School of Engineering, Santa Clara University, Santa Clara, CA 95053, U.S.A.

E-mail: {qle, aamer, jholliday}@scu.edu

Received September 23, 2018; revised March 31, 2019.

Abstract One way to increase storage density is using a shingled magnetic recording (SMR) disk. We propose a novel use of SMR disks with RAID (redundant array of independent disks) arrays, specifically building upon and compared with a basic RAID 4 arrangement. The proposed scheme (called RAID 4SMR) has the potential to improve the performance of a traditional RAID 4 array with SMR disks. Our evaluation shows that compared with the standard RAID 4, when using update in-place in RAID arrays, RAID 4SMR with garbage collection not just can allow the adoption of SMR disks with a reduced performance penalty, but offers a performance improvement of up to 56%.

Keywords disk array, redundant array of independent disks (RAID), shingled magnetic recording (SMR), shingled write

1 Introduction and Related Work

1.1 Introduction

Disk drives have undergone dramatic increases in storage density, totaling over six orders of magnitude in the past five decades; but for capacities to continue to grow, they must overcome a looming physical limit. One of the most promising approaches to overcoming this limit is shingled magnetic recording (SMR) and its follow-up technology, two-dimensional magnetic recording (TDMR). The attractiveness of this approach is that it requires minimal changes to existing magnetic recording technology, and could easily be adopted with essentially the same physical recording mechanisms. Current disks offer recording densities of 400 Gb/in^2, but with shingled writing 1 Tb/in^2 would be considered an achievable goal[1−4]. Since a shingled write disk would, for most tracks, be unable to perform a non-destructive write operation, data layout and management strategies become essential to the smooth adoption of shingled write disks (i.e., without requiring significant changes to overlying software such as file systems, databases, object stores, and logical volume managers). The success of any approach depends on the nature of the block I/O workload presented to the device.

As of 2018, there are many cloud storage providers such as Google Cloud Storage①, Dropbox②, and Backblaze③. There are hundreds of millions of gigabytes of data stored in the mass storage systems used by these providers. These systems need to scale as large as possible while maintaining costs as low as possible. Most of these services target customers using cloud storage as data archiving, which involves mostly writes, less reads, and virtually no updates to archived data. This should be perfect for SMR disks. We propose to use SMR disks to solve the mass storage problem. We have evaluated an array of recorded workloads and their impact on a set of disk models logically representative of the different strategies for data layout on shingled write disks, and proposed a scheme for using SMR disks in RAID (redundant array of independent disks) arrays, named RAID 4SMR. We also evaluated our RAID 4SMR scheme with garbage collection and

Regular Paper

A preliminary version of the paper was published in the Proceedings of MASCOTS 2015.

①Google Cloud Storage. https://cloud.google.com/storage, Jan. 2019.

②DropBox. http://www.dropbox.com, Jan. 2019.

③Backblaze. http://www.backblaze.com, Jan. 2019.

analyzed the reliability of the scheme.

SMR drives could behave differently for write and re-write operations. Write operations when coming to SMR drives will be written sequentially to an arbitrary band. Seagate, one of the most major drive suppliers, published an article④ on how SMR drive works in 2013, explained that any re-write or update existing block requires SMR drives to correct not only the requested data, but essentially all data on the following tracks. Western Digital (WD), another major hard drives manufacturer, in a knowledge base article⑤ enables the customers to help themselves, and states that all physical sectors are written sequentially in a direction radically and are only rewritten after a wrap-around. The write behavior of an SMR drive is also confirmed by Aghayev *et al.*[5] in an effort that combines software and hardware techniques to discover key properties of drive-managed SMR drives.

In the remainder of the paper, we describe our experiences evaluating the behavior of SMR disks when used in an array configuration or when faced with heavily interleaved workloads from multiple sources. While our initial results show a potentially dramatic negative impact when dealing with heavily interleaved workloads, they also demonstrate the positive effect of reducing such interleaving. By rethinking a traditional array layout and redirecting re-write operations to a different drive, we have revised the design of a standard RAID 4 array to not just allow the adaption of SMR disks with a reduced performance penalty, but offer a performance improvement of up to 56%. This can be done by adding a dedicated data HDD and replacing data disks with SMR disks in a basic RAID 4 arrangement. Finally, we propose RAID 4SMR (we change the name from RAID 4S to RAID 4SMR to avoid confusion with RAID 4S by Wacha *et al.*[6]) which explores how SMR disks can be combined with conventional magnetic recording (CMR) or standard disks to achieve an efficient and reliable RAID array. We extend our preliminary work on RAID 4SMR to support garbage collection, followed by detailed analyses on fault tolerance, space efficiency, and reliability.

1.2 Related Work

Wacha *et al.*[6] used the name of RAID 4S for adding faster SSD to RAID arrays to alleviate the RAID 4 parity bottleneck which actually replaces parity disk in RAID 4 with SSD to improve small writes. This is different from our approach which replaces all the standard data HDDs with SMR disks (not just the parity disk). Introducing a RAID 4 array with all SMR disks as data disks would be more challenging since SMR disks are getting worse with a great number of update-in-place operations[7].

Jin *et al.*[8,9] proposed an SMR RAID file system. The difference is that theirs is based on RAID 5 whereas ours is RAID 4. Ours is more extendable, as we can chain multiple RAID 4 systems together. Also, Lu and Zhou[10] tried to employ SMR RAID with SSD. Aghayev *et al.*[5] tried combining software and hardware techniques to reverse engineer key properties of SMR drives.

Earlier work has been described on shingled disks and their physical design, including basic mechanisms to deal with the problem of destructive updates[1−4,11−16]. Most proposed techniques revolve around some form of log-structuring of the data[17−23]. The log-structuring mechanisms owe their designs to the original log-structured file-systems[24−26], and subsequent work applying the same techniques in areas as diverse as databases, tertiary storage, and tape-based filesystems[27−33].

Recent studies describing the management of data on shingled write disks have been described by Gibson and Polte[16], and by Casutto *et al.*[34] The latter offered one of the first practical solutions to managing a log-structured layout in the presence of limited metadata storage capacity, while Amer *et al.*[35,36] explored a spectrum of design parameters for shingled write disks, including alternative interfaces such as object-based stores, or file system based approaches to addressing the new disk behavior.

Also, there are some reliability analyses for different RAID systems which focus on mean time to data loss (MTTDL)[37−41].

Yang *et al.*[42] presented a virtual persistent cache to remedy the long latency behavior of host-aware SMR. He and Du[43] introduced an approach to SMR translation which adapts a drive-managed SMR data management scheme.

④Seagate Technology LLC (2013). Introducing Seagate SMR. https://www.seagate.com/tech-insights/breaking-areal-density-barriers-with-seagate-smr-master-ti, Jan. 2019.

⑤Western Digital. TRIM Command Support. https://support.wdc.com/knowledgebase/answer.aspx?ID=26014, Jan. 2019.

2 Background

Magnetic recording is rapidly approaching a physical limit that cannot be avoided without significant changes to the methods used to record data. The "media trilemma" is a term used by Sann *et al.*[14] to describe the physical limit that hard drives are rapidly approaching, illustrated in Fig.1.

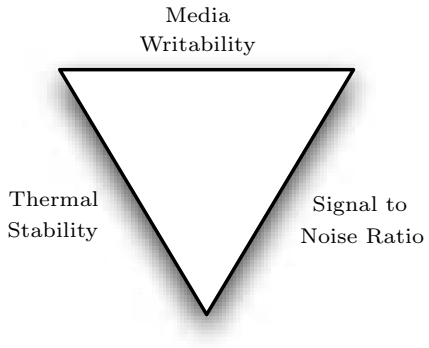

Fig.1. Media trilemma (a term coined by Sann *et al.*[14]).

To overcome the recording limits imposed by the media trilemma, shingled magnetic recording offers a solution that does not require a departure from the basic mechanisms of magnetic recording used in current disks. It requires minor modification to the read-write head in order to allow for "narrower" tracks to be written, and this comes at the cost of a functional difference in how tracks can be updated.

To illustrate how a disk employing shingled magnetic recording can effect increased data storage densities, we start with a logical view of tracks and an illustration of the required modification to the disk head. In Fig.2 we see how data is organized on a disk in a series of adjacent "tracks" each of which is distinctly written and read. The media trilemma dictates a minimum "width" on such tracks. It is important to note that this limit is imposed upon the tracks being written, not when they are read. The writability of the medium (acting to allow reduced widths) coupled with the strength of the magnetic field (acting to widen the track) is in conflict when the track is being written. Assuming we were able to write a narrow track, we would be able to read much thinner tracks than those the trilemma allows us to write. Assisted recording methods exploit this fact, by temporarily overcoming the difficulty of writing to a more stable medium using focused heat (thereby the "wider" field would be used, but the width of the track would be restricted to the area that has been heated). Shingled magnetic recording simply uses a more powerful magnetic field to perform writes, requiring no such assistance, but to allow for narrower tracks, a modified disk head is employed. Specifically, a magnetic shield is added to the trailing sides of the head, as shown in Fig.3(a). In this manner, as a track is written it will be written as a "wide" track. However, such width would only impact the current track but not the entirety of the preceding track as it would be protected by the trailing shield. This would allow us to not only bring tracks closer, but effectively overlap them. Resulting in a shingled track arrangement, where all that is left of a track after the shingling, is what it needed to read the data, not the greater width necessary to initially write the track. In this manner, we get increased disk density primarily through the increase of track density, as illustrated in Fig.3(b). However this increased density comes at the expense of rendering any subsequent attempt to update these narrower preceding tracks destructive.

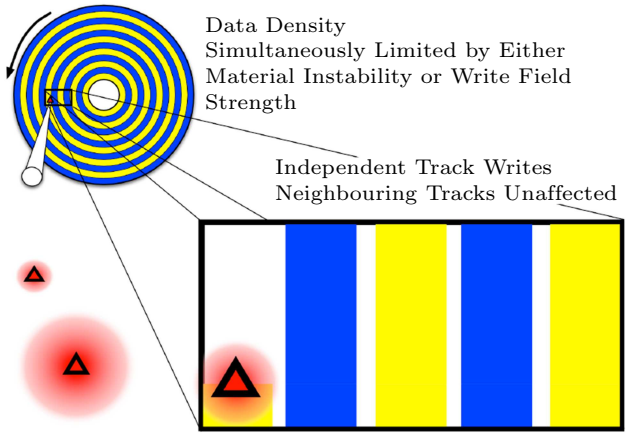

Fig.2. Conceptual view of tracks, as written with a current disk head. Decreasing media writability (to improve stability) for the sake of increasing density would have an adverse effect on track density if the disk head resulted in "wider" tracks.

A disk for which all the tracks are overlapped would likely be impractical for use as a random-access block storage device, and thus a suitable layout scheme for the written blocks and tracks is essential to maintain existing functionality of magnetic hard drives. If a shingled write disk (an SMR drive) were to write all its tracks in a shingled manner, then from one edge of the platter to the other, all tracks would be overlapped. This would mean that updating any previously written track would necessitate pre-reading all adjacent tracks that would be affected, so as to write them back to the disk after updating the desired track. Unfortunately, this would not be limited to a handful of tracks adjacent to

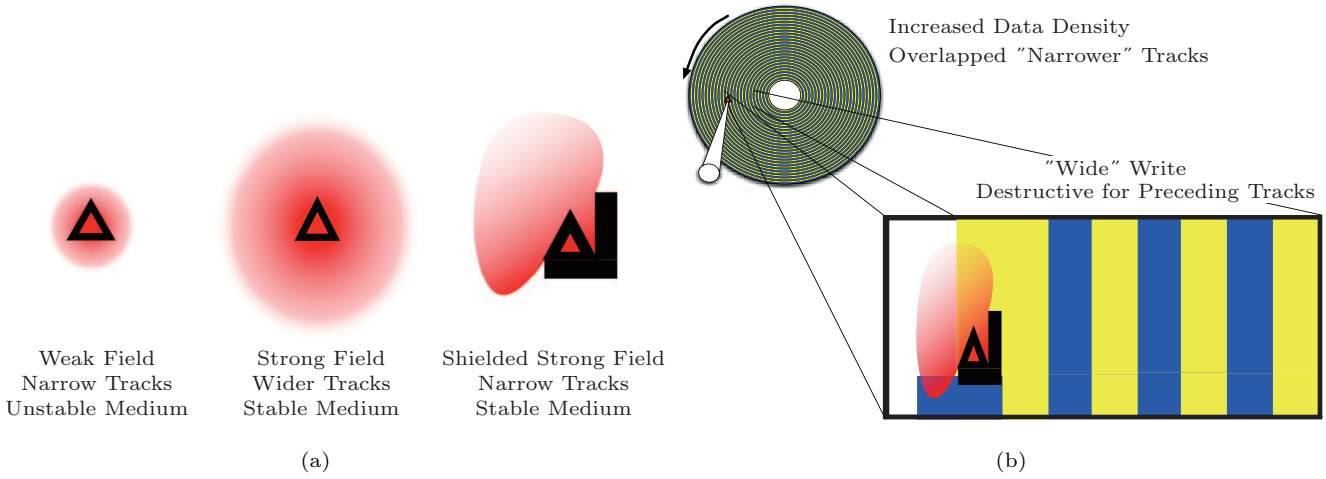

Fig.3. Shingled magnetic recording (SMR), increasing data density through the use of overlapping tracks, written through the use of a shielded disk head. (a) A shielded disk head, with side and trailing edges shielded. This protects tracks written on one side of the head. (b) Increased track and disk density thanks to shingled magnetic recording. Such a shingled-write disk gains storage density by overlapping successively written tracks, leaving "narrower" tracks in its wake.

the track being updated, but as each track would itself have to be written back to the disk, this would result in the need to read further tracks as the neighboring tracks are restored (as restoring each of the neighboring tracks would itself be equivalent to the original request to update the first track). In this manner, any update of an earlier track in its existing location would necessitate re-writing the entire disk if it were completely shingled.

To avoid the need to update a complete disk to accommodate the update of a previously written track, and to effectively localize updates to smaller discrete portions of the disk, a shingled write disk is arranged into distinct bands[11,16,34,35]. We illustrate the logical view of such bands in Fig.4. The number of tracks assigned to a band, the workload observed, and the manner in which written block is handled by the data layout scheme affect the performance of a shingled disk.

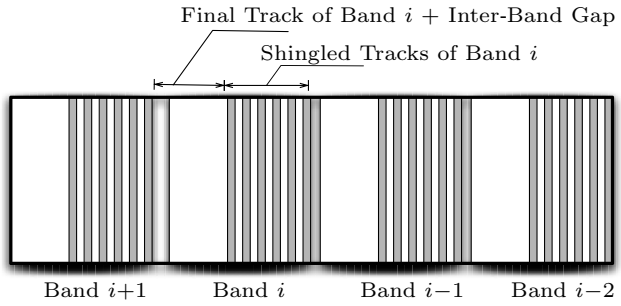

Fig.4. Logical view of a shingled write disk divided into bands, allowing the in-place update of a band, although at the expense of a destructive track write within an individual band.

3 SMR in Server Environments

An initial evaluation of the impact of shingled writing under varying workloads and for different device parameters (band size and buffer capacity) was done in [44]. To evaluate the impact of varied workloads on an SMR drive, we attempt to gather performance metrics that are as universal as possible. We wanted to avoid metrics that depend heavily on physical characteristics of disks, including those yet to be built. We focused on the functional nature of the drives, the shingling of tracks within a band, and the resulting impact on data transfer tasks. While we were investigating the use of LBA distances to evaluate the performance of SMR drives, we found that LBA is unreliable as a metric to analyze movements in SMR drives. For drive-managed SMR drives, to maintain the consistency of data next to the written block and efficiently update data, LBA address may be dynamically mapped to another physical block address (PBA). To this end, we used metrics such as the logical block movements (block distance) instead of time to read/write a block of data. The number of block movements is the difference between the logical address of the first block visited and that of the next block visited. We also collected other logical "movement" metrics, such as track movements (one movement of which results from the need to move the disk head from one track to another) and band movements (when the head moves from one band to another band). Another metric is the number of direction changes (i.e., the number of times the order of access to blocks, tracks, or bands changed). Such direction

changes measure how often the disk head is required to move in another direction or to skip a block/track/band on the move. For example, if the head is reading track 0, and going to track 6 due to a read request, we have one logical track direction change. We collected all of these metrics and found that aside from differences in scale, they are largely correlated. Therefore, our experimental results presented in this paper use the block movements metric.

In Section 4 "Building SMR-Aware Disk Arrays" we propose one possible approach to building an SMR aware disk array suitable for use in a server environment. We, therefore, first evaluate the impact of data placement and benefits or lack of them from interleaving workloads that involve writes originating from multiple sources on these disk arrays. Whether an array of shingled disks is arranged as a simple spanning arrangement, or a striped arrangement (aimed at increasing effective bandwidth), can dramatically affect the amount of data relocation and re-writing required to maintain a shingled write drive. We find that a workload that originates from a heavily interleaved mix of sources is detrimental to shingled write disk performance. We reached these preliminary conclusions through the replay of recorded workload traces.

The workload types collected are block I/O level traces[45] drawn from a variety of system types, block traces reconstructed from web server HTTP request logs⑥, and new block-level traces which we have collected from general file system usage. From both the reconstructed web traces and our own traces, we were able to generate workloads representative of specialized applications. The web traces demonstrate the behavior of a block storage device used to host web pages, while one of our file system traces was drawn from a file system being used to host the image files of a local VMWARE installation.

To evaluate the impact of shingled writing when employed on disks arranged in array, we took the same four recorded workloads and replayed them against a simulated drive to measure the number of track-to-track movements that would be incurred under different conditions. We did not consider disk parallelism in this case to simplify the results.

Fig.5 shows the logical arrangement of blocks we evaluated, while Fig.6 shows a sample of the preliminary results we observed for the amount of inter-track movement resulting from a total of eight different configurations of block arrangement and workload interleaving. All the results in Fig.6 were based on a shingled write disk utilizing a log-structured write scheme to minimize the need to copy overlapped blocks when an in-band update was required. The pure workload shows the total amount of disk activity across four disks arranged in sequence, with workloads replayed sequentially and including no interleaving. In other words, four consecutive traces were each replayed in their entirety, and consecutively, against a disk array employing a spanning layout. This effectively simulated the behavior of a workload that varied over time, but the pure workload at no point included requests interleaved with others of a different workload. The striped workload combines four different workloads, and replays the composite workload against a striped organization of disk blocks across four disks. The workload was generated by randomly interleaving the operations from each of the four workloads in limited bursts. The x-axis of the figure represents each burst size, increasing from a minimum of 1 (where the interleaving is maximized) up to bursts of 1 000 operations. Finally, the dedicated results represent the behavior of the shingled write disks when each disk is dedicated to an individual source workload.

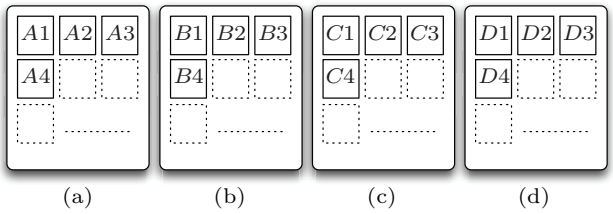

Fig.5. Logical view of a simple array of disks. In the striped arrangement, blocks 0, 1, and 2 are arranged as $A1$, $B1$, and $C1$, respectively. In pure arrangements, blocks 0, 1, and 2 are arranged as $A1$, $A2$, and $A3$, respectively. (a) Disk A. (b) Disk B. (c) Disk C. (D) Disk D.

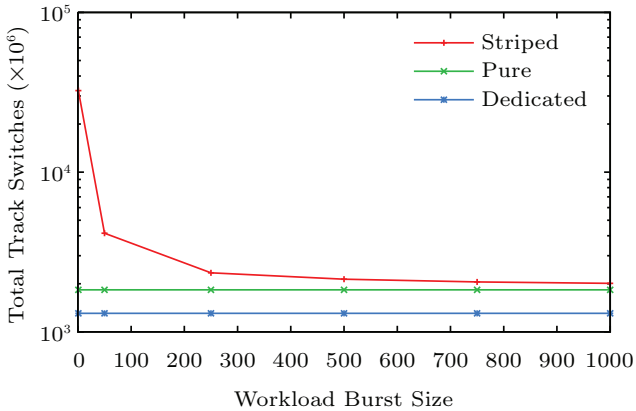

Fig.6. Disk activity when replaying multi-source traces against a simulated array of shingled write disks.

⑥NASA. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, Mar. 2011.

Fig.6 shows that as the degree of interleaving in the composite workload traces is reduced (and the burst sizes increase) for the array, we see a reduction in the amount of disk activity that approaches that of the pure configuration. This is predictable and expected, as replaying a sequence of traces without any interleaving is exactly what is done by that configuration, and is the ultimate destination of extending burst sizes until they encompass an individual workload trace in its entirety. The surprising observations are just how much more activity results when unrelated operations are finely merged into a composite trace, and how further improvement can be achieved by separating workloads from different sources to individual dedicated disks. For a workload created from interleaving operations from multiple sources into small bursts, the amount of movement caused by relocating disk bands rises dramatically (up to 40 times in this instance, though quickly dropping as the burst size increases to the level of 50 and 250 operations per burst). We attribute this behavior to the increased likelihood of unrelated data being written in adjacent positions increasing the likelihood of an update being required that is unrelated to much of the data on the same band. This problem is alleviated as the burst sizes increase, and eliminated entirely when individual dedicated disks are used. The difference in the dedicated configuration is that, unlike the pure configuration, it will never result in the writing of data from different data sources to the same device. Because the dedicated configuration avoids this risk entirely, we see a further drop in disk activity of around 25%. Based on these observations, in Section 4, "Building SMR-Aware Disk Arrays", we will propose RAID 4SMR which is a disk array system based on RAID 4[46] using SMR disks.

4 Building SMR-Aware Disk Arrays

In computer storage, there are several different standard and non-standard RAID (redundant array of independent disks) configurations such as RAID 0, 1, 2, 3, 4, 5, 6, 10. These configurations help to build a large reliable storage system from more than two common hard drives. This can be done by using one or more of the techniques of striping, mirroring, or parity. In this manuscript, we choose to focus our study around RAID 4, instead of RAID 5 (distributed parity blocks) or RAID 6 (extra parity blocks), because it is the simplest and lowest overhead version of parity-based RAID, and therefore allows us to evaluate the impact of SMR integration most cleanly (i.e., without introducing additional variables that are tangential to the question of SMR's impact). When we structure data appropriately in the log fashion, there is no advantage of RAID 5 over RAID 4. RAID 6 deals with multi-disk failure which creates more overhead compared with RAID 4.

Shingled write disk with the SMR technology can help us triple data density in the future[2], but it comes with a price of update in-place. The degradation of performance gets worse if we just switch regular HDDs to SMRs in RAID arrays[7]. We now propose RAID 4SMR in Fig.7 as an approach to utilizing SMR devices in a disk array arrangement which is a hybrid system of three SMRs and two HDDs for a redundant array. We have chosen a scheme that maintains a traditional block interface so that an SMR device will integrate easily into existing storage architectures. On the SMR disks we eliminate the update-in-place operation due to the high cost of updating. The regular HDD has the advantage of in-place update efficiency; but as we are not limited to traditional hard drives, we can use traditional HDDs or recently popular SSDs. The mapping table can be easily stored in battery-backed memory (called NVRAM).

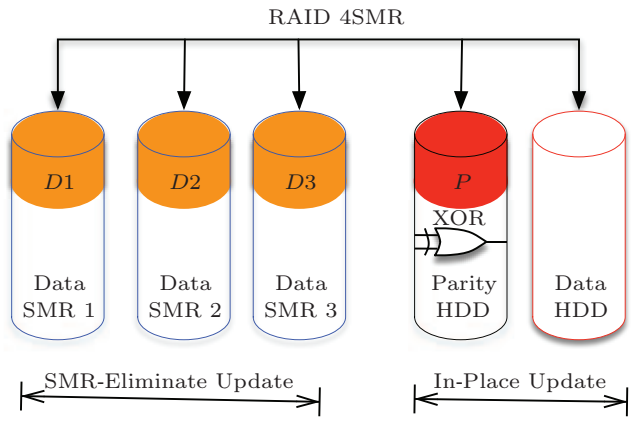

Fig.7. When the data is first written to the array, parity disk will store P = XOR($D1$, $D2$, $D3$, $DataHDD$). Data HDD blocks are expected to be zero-initialized.

When dealing with garbage collection, only actively updated blocks are tracked in a hash table along with a location of the actual block in Data HDD. If a band with a block number is in the mapping table, it will be considered as an invalid or dirty block. In addition to the mapping table, a simple lookup table for a number of invalid blocks in a band is maintained to quickly identify when the garbage collection procedure will be triggered (which reduces the number of active

re-mappings that need to be tracked). Together, only modest mapping and lookup tables are necessary to retrieve the correct data.

Since SMR is the most suitable for archival storage or a WORM (Write One Read Many) disk, we design our SMR RAID to be a solution for reliable data storage arrays. The reason why we chose RAID 4 instead of other RAIDs is that RAID 4 stores frequently updated parity blocks on a dedicated disk, and the rest three data disks can be all replaced by SMR disks.

One of the advantages of the RAID 4SMR design is that we can chain many RAID 4SMR systems together with one large Data HDD (or SSD to avoid performance bottleneck) as shown in Fig.8. The Data HDD does not need to be the same size with other SMRs and the Parity HDD. This scheme is suitable for the enterprise level where a lot of the data is archival. All the data in the system is protected by one or more parity disks in chaining systems.

Fig.8. RAID 4SMR can be chained together to form a bigger array.

We anticipate that the volume size for this simple standalone RAID 4SMR would be around 30 TB–40 TB. This could be accomplished with an array of 3×10 TB SMR disks and 2×10 TB regular HDD disks (for both parity and data disks).

In Fig.7 we illustrate our approach to maintaining a block interface for a disk array built around shingled write devices. In this example, data is held primarily on disks $D1$–$D3$, which are all SMR devices, while parity is held on disk P, which is not a shingled write disk, but may be composed of one or more traditional magnetic disks, or a device built around a storage class memory technology or flash-based SSDs (recommended). The system is augmented with an additional mass storage device (labeled Data HDD in the figure). This last device can be a traditional magnetic disk with update-in-place efficiency. Its purpose is to serve as a collection of updated data blocks. With this design, Data SMR disks ($D1$–$D3$) only update whole bands during garbage collection. However, this last device need not be a different storage technology but could utilize shingled writing if it serves as a journal to hold each block update.

With an updating intensive workload, all the updates will be redirected to the Data HDD so that the Data HDD now becomes the potential performance bottleneck as well as the Parity HDD disk. We recommend a high-performance disk used in this case.

Since the Data HDD can be updated in an update in-place fashion, and will only serve to hold updated data blocks that could not be updated in-place on Data SMR disks, it will hold a very small fraction of the data in the entire array. But as it needs to be protected against single-device failures, it requires that the parity disk be capable of efficient updates in-place. The parity will need to be updated with every write to the Data SMR disks, or any write that is redirected to this Data HDD (as illustrated in Fig.9). This is why we require the parity disk to be capable of efficient in-place updates. In prior work, we have found that utilizing a hybrid arrangement of data and parity disks, with the parity disk employing a different storage technology, offers performance and reliability benefits for the overall system[47].

While such an architecture might seem to impose a heavy burden on storage capacity, as it appears to require an additional device for every RAID-like storage array, this is not the case in practice. If we estimate that fewer than 5% of all disk blocks are ever updated in most mass storage scenarios, then we can see that the capacity of Data HDD disks will go largely unused. It is therefore possible to utilize the same Data HDD device with multiple RAID 4SMR arrays, as illustrated in Fig.8. This will have a slightly negative impact on overall system reliability, as it creates an interdepen-

dency among up to 20 different arrays (estimate 5% updates) that could potentially be linked in this manner. However, it is important to point out that this is a very minor impact, as each and every array would still be capable of surviving the loss of an individual disk device (including the shared device, for which different portions of its data are necessary for, and dependent on, different arrays).

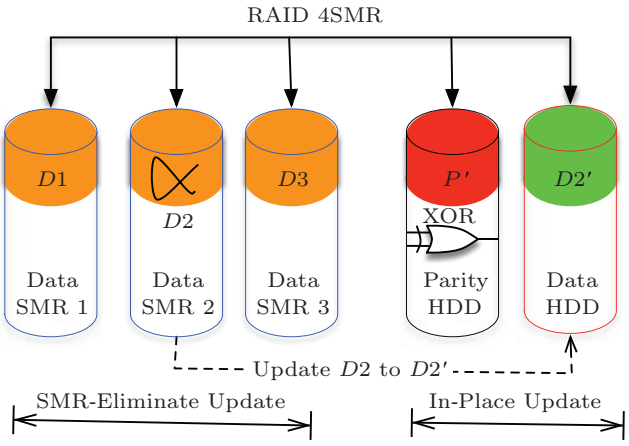

Fig.9. When one of the blocks in the shingled array is updated, the data on SMR disks will be left unchanged, but the updated block will be written into Data HDD and the corresponding parity block is recalculated.

When chaining several RAID 4SMR systems together, we need a scheduling policy to choose which RAID 4SMR disk groups to write. Choosing the right algorithm will also affect the performance when data is retrieved later. In general cases, we can implement a simple round-robin (RR) algorithm to more evenly distribute the burden across disk groups. If one decided to deploy RAID 4SMR disk groups over a complex network fabric, a shortest path algorithm might be a way to improve the load as well.

Assuming a read is randomly distributed over the range of all blocks and we anticipate that 5% of write-operations are rewritten blocks. In the worst case, all updated blocks are in Data HDD (without garbage collection). That means the Data HDD will only be read 5/100 (or 1/20) of the time. In the case of 20 chained RAID 4SMR subsystems, randomly distributed read is going to put the whole system at its highest read performance. It will demand the highest read performance from the Data HDD to avoid performance bottleneck. Because the Data HDD will only be read 1/20 of the time, the read speed of the Data HDD should be greater than the read performance of a standalone RAID 4SMR. Typical read speed for a currently available SMR disk is around 150 MB/s (Seagate 8 TB SMR drive[⑦]). Since RAID 4SMR data is distributed over three Data SMR disks, the expected read speed of a RAID 4SMR subsystem is 3×150 MB/s or 450 MB/s. With a latest SSD disk (the read speed is around 500 MB/s — Samsung SSD 850 EVO[⑧]) used in place of the Data HDD disk (which we recommended), the Data HDD should not be a performance bottleneck.

In Fig.10 we show how RAID 4SMR works when operation requests arrived. If the operation is read (R), we just look up the mapping table and retrieve the data from the appropriate disk and block. The read operation can read from any Data SMR[1−3] or Data HDD disk. If the operation is write (W), we need to figure out whether the write operation is first written to a unique block or update data of a block. If it is the first time that we write to the block, we can simply write data to the next available block in one of the Data SMRs (D_1, D_2, D_3) (the SMR disk only appends a block to a band to avoid destroying data blocks next to it). The controller can write a single block instead of stripe-distributed blocks across all data disks. Because of this, RAID 4SMR can be deployed in a fabric over the network if needed. If the operation is intended to update a block of a Data SMR disk (D_1, D_2, or D_3), we now redirect the write to the Data HDD and mark the stale block invalid in the mapping table. In case the total number of invalid blocks in the band across three data SMR disks is more than pre-defined variable threshold i, the garbage collection process is triggered. The garbage collection process will read valid data blocks in bands from across three data SMRs (D_1, D_2, D_3) and the Data HDD disks and rewrite the bands with valid blocks, new block, and updated blocks in the Data HDD. Also, the mapping table will be updated.

4.1 RAID 4SMR Fault Tolerance

Every hard drive fails eventually. Both RAID 4SMR and RAID 4 have a fault tolerance of one drive. RAID 4SMR can survive one drive failure in any drive. This is guaranteed to work since RAID 4SMR does not distribute blocks in a stripe across all data disks. Instead, it will rely on the controller to only deal with a single block. The controller with a mapping table

⑦Seagate Technology — ST8000AS0002. https://www.seagate.com/www-content/product-content/hdd-fam/seagate-archive-hdd/en-us/docs/archive-hdd-ds1834-5c-1508us.pdf, Jan. 2019.

⑧Samsung Electronics Co., Ltd. https://www.samsung.com/semiconductor/minisite/ssd/product/consumer/850evo, Jan. 2019.

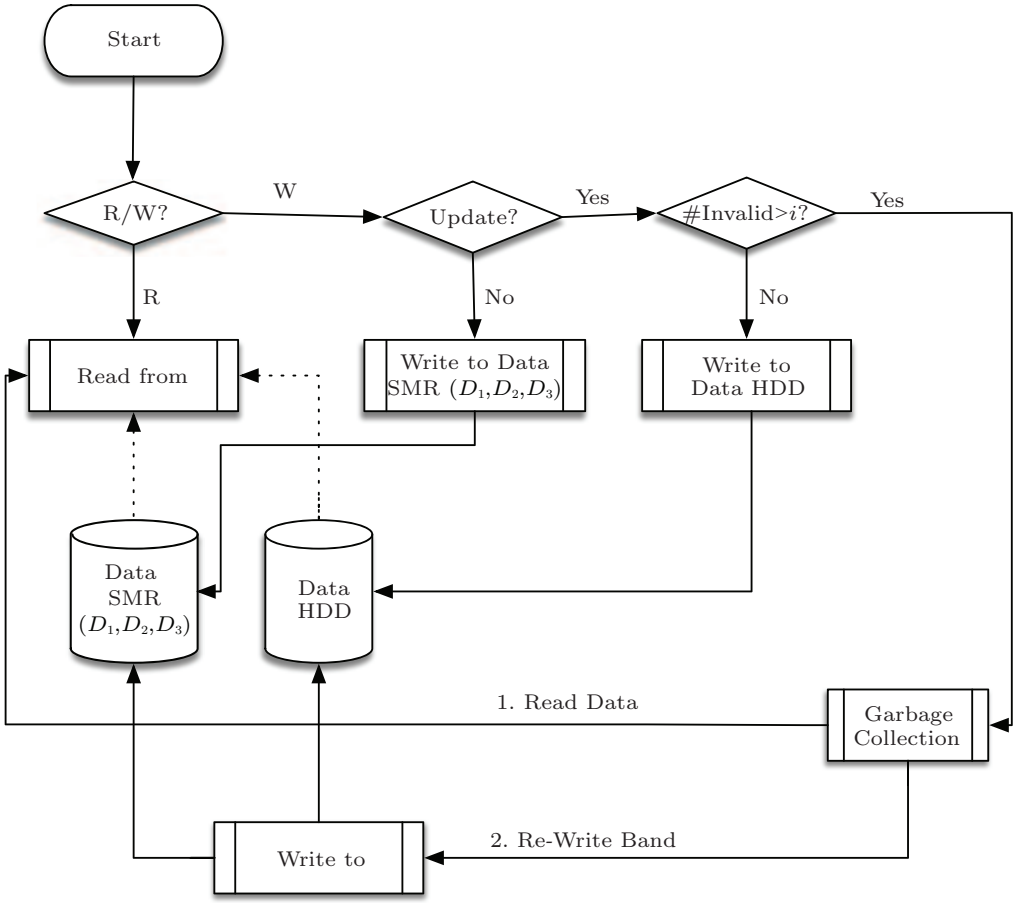

Fig.10. Flowchart of how RAID 4SMR works.

will be able to collect all required blocks to get the data back. This mechanism even works when we update more blocks in the same stripe which are eventually written to the Data HDD. At any time, all blocks can be retrieved to get the data back, and any block in a RAID 4SMR system will be protected with a parity block. When one of the disks fails, the array is in degraded mode and the failed drive needs to be replaced. The repair procedure is much the same as that for RAID 4.

Parity bit in RAID 4SMR is calculated as $P = D1 \oplus D2 \oplus D3 \oplus DataHDD$. If the failed drive is one of the data disks, we can calculate the lost value based on the parity disk and the other three online disks. If the failed drive is the parity disk, we just recalculate the parity value again. If the failed drive is the Data HDD, we can also calculate the lost value as we have the parity disk and other online disks.

In the design of RAID 4SMR, we maintain the reliability of RAID 4, in which the array can tolerate one drive failure (any drive, including the crucial Data HDD). In case of updated blocks, data is redirected to Data HDD first before being written back to a Data SMR in the garbage collection process. When a block is written to Data HDD, the related parity block is also updated with the new parity value $P = D1 \oplus D2 \oplus D3 \oplus DataHDD$. The new parity value ensures the new block written to the Data HDD is still protected. These new parity values also guarantee all the blocks in the Data HDD are protected with other SMR disks and the parity disk. In case the crucial Data HDD fails, the data in it can be restored from the other three Data SMRs and the parity HDD.

4.2 RAID 4SMR Space Efficiency

Space efficiency is the fraction of the total drives' capacity that is available to use for data (as opposed to parity) blocks. The expression has a value between 0 and 1. The higher value is the better.

Let n represent the number of disks in an array. Standard RAID 4 (with $n = 4$) has a space efficiency of $1 - \frac{1}{4} = 75\%$ because data is distributed over three

disks and one of the four disks is used as a parity disk.

Unlike standard RAIDs, RAID 4SMR space efficiency varies widely depending on the number of its updated blocks. Archival data which is never rewritten is actually not taking advantage of the dedicated Data HDD. In this worst case, the data blocks can all be stored on three SMR disks (without update) in an array of five disks. The best case is when all the data SMRs are filled up and the Data HDD is also filled up with updated blocks. In this case, data is stored on four disks (all three SMRs and the Data HDD) and only the parity disk counts against the space efficiency. The space efficiency for RAID 4SMR is in this range:

$$1 - \frac{1+c}{n} \leqslant \text{space efficiency} \leqslant 1 - \frac{c}{n},$$

where n is the number of disks, and c is the number of chaining systems in RAID 4SMR array. With $n = 5$ and $c = 1$ (simple standalone RAID 4SMR array), the efficiency has a range from 60% to 80%.

Let u represent the percent of updated blocks. Before the garbage collection, the space efficiency is supposed to be lower at u percent as we do not have to reclaim those invalid blocks.

When chaining multiple sub RAID 4SMR systems to take advantage of Data HDD and form a massive storage system (Fig.8), we anticipate the efficiency is in line with RAID 4. Let us say we are chaining 20 RAID 4SMR systems, n will be 81 (four disks for each RAID 4SMR system and only one Data HDD needed), c will be 20, and the efficiency has a range from 74% to 75%.

4.3 RAID 4SMR Reliability

In this subsection we evaluate the long-term reliability of a simple standalone RAID 4SMR disk array consisting of four data disks and a parity disk. The reliability of RAID 4SMR is different than that of standard RAID 4. The most popular way to estimate the reliability of a redundant disk array is using mean time to data loss (MTTDL). When a disk fails, the repair process is triggered immediately. Let us assume that disk failures are independent events, exponentially distributed, denoted by λ as failure rate. The repair is exponentially distributed with rate μ.

To simplify the calculation, we analyze a simple RAID 4SMR array with five disks. The Markov Chain diagram in Fig.11 displays the simplified state transition probability for a RAID 4SMR array without chaining. State $<0>$ is the ideal state which represents the normal state of the array when all five disks are operational as expected. The starting state is state $< 0 >$ (safe), from which we transition to state $< 1 >$ (degraded) at rate 5λ whenever one of the five disks fails. As we know, RAID 4SMR can recover from one disk failure, and a failure of a second disk would bring the array to data loss state.

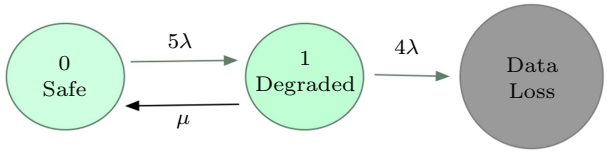

Fig.11. State transition probability diagram — Markov chain for RAID 4SMR.

The Kolmogorov system of differential equations describing the behavior of the RAID 4SMR array has the form:

$$\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} = -5\lambda p_0(t) + \mu p_1(t),$$
$$\frac{\mathrm{d}p_1(t)}{\mathrm{d}t} = 5\lambda p_0(t) - (4\lambda + \mu) p_1(t),$$

where $p_i(t)$ is the probability that the system is in state $<i>$ at time t with the initial conditions $p_0(0) = 1$ and $p_1(0) = 0$.

The Laplace transforms of these equations are:

$$sp_0^*(s) = -5\lambda p_0^*(s) + \mu p_1^*(s) + 1,$$
$$sp_1^*(s) = 5\lambda p_0^*(s) - (4\lambda + \mu) p_1^*(s).$$

Observing that the mean time to data loss (MTTDL) of the array is given by:

$$MTTDL = \sum_i p_i^*(0).$$

We solve the system of Laplace transforms for $s = 0$ and use this result to obtain the MTTDL of the array:

$$MTTDL = \frac{\mu + 9\lambda}{20\lambda^2},$$

with mean time to failure (MTTF) and mean time to repair (MTTR) defined as:

$$MTTF = \frac{1}{\lambda},$$
$$MTTR = \frac{1}{\mu}.$$

4.4 RAID 4SMR with Garbage Collection Evaluation

We showed how RAID 4SMR[48] works without garbage collection, which is an ideal situation as we do not have to reclaim the freed blocks. In this subsection, we evaluate the same RAID 4SMR with garbage collection in our simulation to determine the trade-off of using an SMR disk in RAID 4SMR vs a standard RAID 4 array. Below is the simple pseudo code for the garbage collection algorithm.

```
if totalInvalidBlocksInBands() >pre-defined-i then
    buf ← getData(DataSMR(D1, D2, D3), DataHDD);
    deleteOldBlocksInDataHDD(); // blocks stored in
    buf;
    rewriteSMRBands(buf);
end
```

In order to perform garbage collection, we add the support for reclaiming freed blocks in our simulation. This garbage collection process will be triggered when we have enough i freed blocks, which is a pre-defined value. Value of i is in the range of $0 < i < 3 \times band\_size$. The highest value means all blocks in three bands across the three Data SMR disks are invalid. In the simulation, i is chosen as the number of blocks in a band (or band size). It means that when the garbage collection is triggered, i blocks of updated blocks are read from the Data HDD. These blocks will replace invalid blocks in related bands across three SMR disks. After that, these all valid bands will be written back to three SMR disks. The bands are now fully updated with valid blocks.

Using the collection of workloads we gathered, we mix those workloads in the same category together to form newly-merged workloads to simulate the multi-user environments. Although these new workloads have the same name, they may not have the same characteristics as standalone workloads. We have plotted the new workload's percentage of written and updated blocks in Fig.12. We define an update as an operation attempting to write to a block that was previously written during the observation period. The update percent is the percentage of total blocks that were updated. The number on the x-axis is the number of write operations in the case of the write percent and the number of update operations for the update percent. Workloads show a variation of the percentage of updates across and within workload types.

As an SMR drive cannot simply perform an update in-place, we paid particular attention to block update operations. It is these operations that, unless they happen to fall at the last track of a band, would require us to address the update-in-place restriction. Our analysis differs from prior art[35] in that we do not track updates as simply blocks that were written more than twice as a percentage of the write operations experienced by the device. We plot the percentage of blocks that were observed to be writes or updates. At each data point, we plot the percentage of all blocks that were written

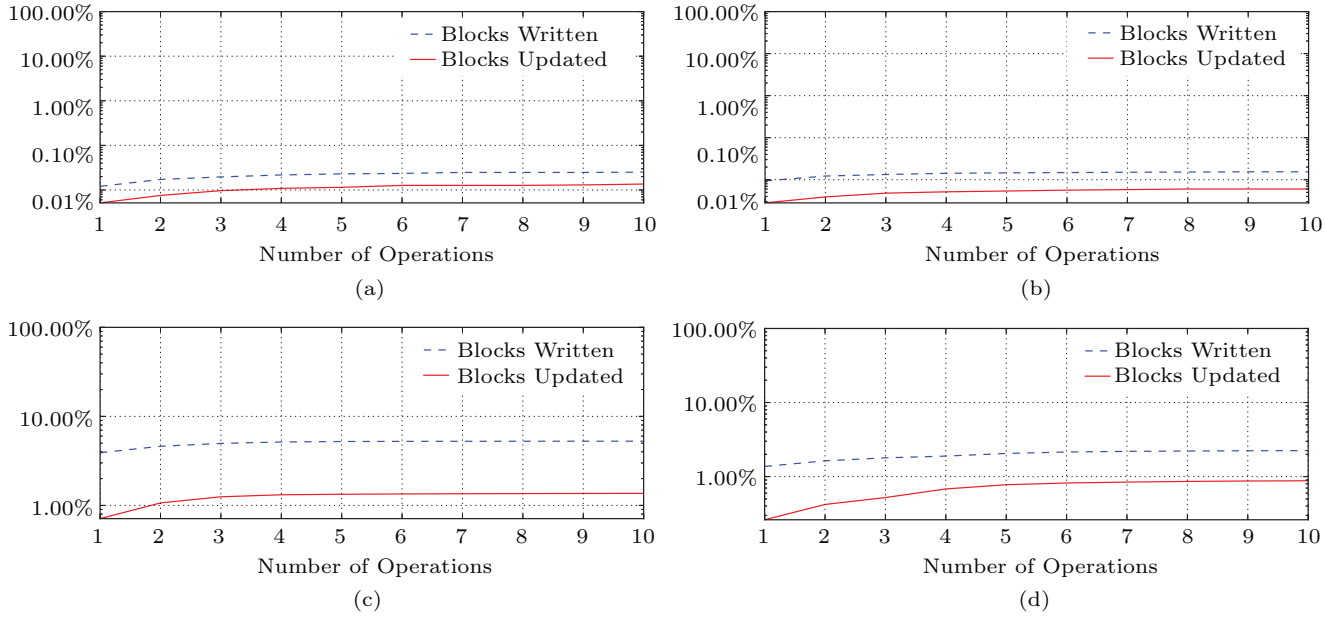

Fig.12. Four merged workloads from the same category. (a) NASA: 0.031% writes, 0.019% updates. (b) RSRCH: 0.016% writes, 0.006% updates. (c) VMWare: 5.286% writes, 1.382% updates. (d) WEB: 2.387% writes, 1.016% updates.

at most x times, where x is given on the x-axis as a percentage of all blocks. Similarly we also plot the percentage of all blocks that were updated at most x times as a percentage of all blocks. These quantities are related, but while the percentage of blocks updated at most x times might appear to be the same as the percentage of blocks written at most $x + 1$ times, that is not necessarily the case.

The y-axis is presented in log-scale. Same generated workloads (NASA, RSRCH, VMWare, and WEB) will be used for all RAID 4, RAID 4SMR, and RAID 4SMR with garbage collection. With that, we can calculate the overhead of garbage collection in the same RAID 4SMR systems as well as RAID 4 with SMR disks. In this subsection, we will evaluate the use of an SMR disk in a RAID 4SMR array with garbage collection.

If a RAID 4SMR uses a parity bit to protect the array from a drive failure, it will cost more for a write intensive system. In the best case scenario, we can expect blocks are distributed all over the data disks, which means the maximum of read/write performance is $(n - c - 1)X$.

If the array is in degraded mode, it will affect the system performance since all the related disks (though not the replaced data disk) will need to be read in order to get the data back. Since most of the data on the SMR disk is archival data, the overall system performance will not be impacted so much as that with regular RAID arrays where data is accessed/updated frequently. The performance of the RAID arrays, in any case, is only restored when a new SMR/HDD is replaced and data is re-synced. The process to re-sync a 4 TB data disk nowadays can take more than 10 hours in a hardware RAID system. This is even worse if the scheme is implemented in a software RAID array.

To evaluate the efficiency of RAID 4SMR over RAID 4, we calculate the ratio of RAID 4SMR over RAID 4 (by the number of block movements as usual). To be fair, the number of movements only counts for the first three data disks in both RAID 4 and RAID 4SMR (parity and Data HDD disks movement do not count) because the standard RAID 4 has only four disks. The result table's (Table 1) last column shows that RAID 4SMR always performs better than standard RAID 4 thanks to the deferring of update operations to the Data HDD in RAID 4SMR. In the case of RSRCH mixed workload with extremely low write and update percents, we can see a slight improvement of 1.5%. This improvement can go up to 56% in our evaluation with WEB mixed workload (high write and update percents).

When adding garbage collection to a RAID 4SMR scheme, we expect to see some overhead for garbage collection operations. This overhead should not add a tremendous amount of extra block movements. By calculating the ratio of RAID 4SMR with garbage collection (RAID 4SMR gc) and RAID 4SMR (Table 1) in the first column, we are confident that RAID 4SMR gc is usable in the real world with different types of workloads. The maximum overhead of 2.11% in the case of a VMWare mixed workload (5.28% written and 1.38% updated) is very low. RAID 4SMR with garbage collection is ideal for archival workloads which have almost no overhead for garbage collection and less block movements compared with standard RAID 4. While deploying SMR disks in a chain of RAID 4SMR, the benefits will be even more pronounced since space efficiency can max out at 80% compared with the 75% of a standard RAID 4. In general, RAID 4SMR (with or without garbage collection) performs better than standard RAID 4 with fewer block movements.

Table 1. Ratio of Number of Block Movements for RAID 4, RAID 4SMR, and RAID 4SMR gc with SMR Disk

Workload	RAID 4SMR gc/ RAID 4SMR (%)	RAID 4SMR gc/ RAID 4 (%)	RAID 4SMR/ RAID 4 (%)
NASA	100.00	97.72	97.72
RSRCH	100.00	98.57	98.57
VMWare	102.11	52.52	51.44
WEB	100.78	44.66	44.32

5 Conclusions

We started out evaluating an SMR disk with our simulation with many different traces. Later, we ran the simulation with SMR disks in array configurations. We found that without a proper scheme, SMR disks are almost unusable in an array configuration.

In this manuscript, we proposed RAID 4SMR with SMR disks in places of data disks in a RAID 4 array. Contrary to the popular belief that an SMR drive is only suitable for archival storage and would perform worse in RAID arrays, our evaluation showed that with proper design modifications, such as our proposed RAID 4SMR, SMR disks can be effectively employed in place of standard HDDs. With RAID 4SMR, SMR drives can greatly improve not just data density, but also performance, while maintaining the same reliability. In other words, an appropriately SMR-aware arrangement, like RAID 4SMR, allows the SMR disk to

be more effectively used for mass storage systems or over network fabrics.

With SMR disks, it is easy to assume that any workload that is heavily biased toward writes would be problematic. Our workload analysis has shown the opposite. Considering the percentage of blocks that experience updates is at least as important as taking into account the number that experience writes. In other words, a single write not to be repeated at the same location is very different from multiple writes to the same location at different times. We used a logical distance metric based on the number of block movements, which is independent from request timings and changes in precise physical performance characteristics of devices.

While our previous results in [44] show a negative impact when dealing with heavily interleaved workloads, they also demonstrate the positive affect of reducing such interleaving. This can be achieved either by rethinking a traditional array layout and dedicating disks and bands, or by directing independent workloads to different devices/bands. Our proposed RAID 4SMR with garbage collection scheme clearly demonstrated the feasibility of using SMR disks in a RAID 4 array by outperforming the use of SMR disks in a standard RAID 4 with update in-place by 56%. Directing different workloads to different devices can be aided by existing efforts on workload differentiation and tagging[49], and would be a simple way to avoid heavily interleaved workloads.

We started with RAID 4 because it is the simplest and lowest overhead version of parity-based RAID, and best allows us to focus on the impacts of using SMR in generic RAID arrangements. Other standard RAID arrays such as RAID 5 and RAID 6 come with some unique characteristics which require more detailed and focused studies to evaluate the impact of SMR in more specific contexts. Increasing the reliability of RAID 4SMR can also be done by using extra parity disks in a two-dimensional RAID array[50].

For future work, a kernel driver for this RAID 4SMR with garbage collection is needed to adapt the design to the real world. Also, a fabric that interconnects to the individual drives using erasure coding techniques is considered for new development since disk performance is quite unpredictable in large-scale data centers.

References

[1] Shiroishi Y, Fukuda K, Tagawa I, Takenoiri S, Tanaka H, Yoshikawa N. Future options for HDD storage. *IEEE Transactions on Magnetics*, 2009, 45(10): 3816-3822.

[2] Tagawa I, Williams M. High density data-storage using shingle-write. http://www.intermagconference.com/intermag2009/src/Program1.pdf, May 2019.

[3] Kryder M, Kim C. After hard drives — What comes next? *IEEE Transactions on Magnetics*, 2009, 45(10): 3406-3413.

[4] Greaves S, Kanai Y, Muraoka H. Shingled recording for 2–3 Tbit/in^2. *IEEE Transactions on Magnetics*, 2009, 45(10): 3823-3829.

[5] Aghayev A, Shafaei S, Desnoyers P. Skylight — A window on shingled disk operation. *ACM Transactions on Storage*, 2015, 11(4): Article No. 16.

[6] Wacha R, Brandt Sent J, Maltzahn C. RAID4S: Adding SSDs to RAID arrays. https://users.soe.ucsc.edu/∼carlosm/Papers/S11.pdf, May 2019.

[7] Le Q, Holliday J, Amer A. The peril and promise of shingled disk arrays: How to avoid two disks being worse than one. In *Proc. Poster Session at the 10th USENIX Conference on File and Storage Technologies*, Feb. 2012.

[8] Jin C, Xi W, Ching Z, Huo F, Lim C. HiSMRfs: A high performance file system for shingled storage array. In *Proc. the 30th IEEE Symposium on Mass Storage Systems and Technologies*, Jun. 2014, Article No. 1.

[9] Liu W, Feng D, Zeng L, Chen J. Understanding the SWD-based RAID System. In *Proc. the 2014 International Conference on Cloud Computing and Big Data*, November 2014, pp.175-181.

[10] Lu Z, Zhou G. Design and implementation of hybrid shingled recording RAID system. In *Proc. the 14th IEEE Int. Conf. Dependable, Autonomic and Secure Computing, 14th Int. Conf. Pervasive Intelligence and Computing, 2nd Int. Conf. Big Data Intelligence and Computing and Cyber Science and Technology Congress*, August 2016, pp.937-942.

[11] Kasiraj P, New R, Souza J, Williams M. System and method for writing data to dedicated bands of a hard disk drive. United States Patent 7490212, 2009. http://www.freepatentsonline.com/7490212.html, March 2019.

[12] Krishnan A, Radhakrishnan R, Vasic B. LDPC decoding strategies for two-dimensional magnetic recording. In *Proc. the 2009 Global Communications Conference*, Nov. 2009, Article No. 606.

[13] Krishnan A, Radhakrishnan R, Vasic B, Kavcik A, Ryan W, Erden F. 2-D magnetic recording: Read channel modeling and detection. *IEEE International Magnetics Conference*, 2009, 45(10): 3830-3836.

[14] Chan S K, Radhakrishnan R, Eason K, Elidrissi R, Miles J, Vasic B, Krishnan A. Channel models and detectors for two-dimensional magnetic recording. *IEEE Transactions on Magnetics*, 2010, 46(3): 804-811.

[15] Wu Y, O'Sullivan J, Singla N, Indeck R. Iterative detection and decoding for separable two-dimensional intersymbol interference. *IEEE Transactions on Magnetics*, 2003, 39(4): 2115-2120.

[16] Gibson G, Polte M. Directions for shingled-write and two-dimensional magnetic recording system architectures: Synergies with solid-state disks. Technical Report, Carnegie Mellon University Parallel Data Lab, 2009. http://www.pdl.cmu.edu/PDL-FTP/PDSI/CMU-PDL-09-104.pdf, March 2019.

[17] Kadekodi S, Pimpale S, Gibson G. Caveat-scriptor: Write anywhere shingled disks. In *Proc. the 7th USENIX Workshop on Hot Topics in Storage and File Systems*, July 2015, Article No. 16.

[18] Pease D, Amir A, Real L, Biskeborn B, Richmond M. The linear tape file system. In *Proc. the 26th IEEE Symposium on Mass Storage Systems and Technology*, May 2010, Article No. 8.

[19] Zhang X, Du D, Hughes J, Kavuri R. HPTFS: A high performance tape file system. In *Proc. the 14th NASA Goddard Conference on Mass Storage Systems and Technologies, the 23rd IEEE Symposium on Mass Storage Systems*, May 2006.

[20] Lin C, Park D, He W, Du D. H-SWD: Incorporating hot data identification into shingled write disks. In *Proc. the 20th IEEE International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, August 2012, pp.321-330.

[21] Moal L D, Bandic Z, Guyot C. Shingled file system host-side management of Shingled Magnetic Recording disks. In *Proc. the 2012 IEEE International Conference on Consumer Electronics*, January 2012, pp.425-426.

[22] He W, Du D. Novel address mappings for shingled write disks. In *Proc. the 6th USENIX Workshop on Hot Topics in Storage and File Systems*, June 2014, Article No. 6.

[23] Hall D, Marcos J, Coker J. Data handling algorithms for autonomous shingled magnetic recording HDDs. *IEEE Transactions on Magnetics*, 2012, 48(5): 1777-1781.

[24] Ousterhout J, Douglis F. Beating the I/O bottleneck: A case for log-structured file systems. *SIGOPS Operating Systems Review*, 1989, 23(1): 11-28.

[25] Rosenblum M. The design and implementation of a log-structured file system [Ph.D. Thesis]. UC Berkeley, 1992.

[26] Rosenblum M, Ousterhout J. The design and implementation of a log-structured file system. In *Proc. the 13th ACM Symposium on Operating System Principles*, October 1991, pp.1-15.

[27] Kohl J, Staelin C, Stonebraker M. HighLight: Using a log-structured file system for tertiary storage management. In *Proc. the 1993 USENIX Winter Technical Conference*, January 1993, pp.435-448.

[28] Selzer M, Bostic K, McKusick M, Staelin C. An implementation of a log-structured file system for UNIX. In *Proc. the 1993 USENIX Winter Technical Conference*, January 1993, pp.307-326.

[29] Dai H, Neufeld M, Han R. ELF: An efficient log-structured flash file system for micro sensor nodes. In *Proc. the 2nd International Conference on Embedded Networked Sensor Systems*, November 2004, pp.176-187.

[30] Finlayson R, Cheriton D. Log files: An extended file service exploiting write-once storage. In *Proc. the 11th ACM Symposium on Operating Systems Principles*, November 1987, pp.139-148.

[31] Lee S, Moon B. Design of flash-based DBMS: An in-page logging approach. In *Proc. the 2007 ACM SIGMOD International Conference on Management of Data*, June 2007, pp.55-66.

[32] Lomet D. The case for log structuring in database systems. In *Proc. the 6th International Workshop on High Performance Transaction Systems*, September 1995, pp.136-140.

[33] Neefe J, Roselli D, Costello A, Wang R, Anderson T. Improving the performance of log-structured file systems with adaptive methods. In *Proc. the 16th ACM Symposium on Operating Systems Principles*, October 1997, pp.238-251.

[34] Casutto Y, Sanvido M, Guyot C, Hall D, Bandic Z. Indirection systems for shingled-recording disk drives. In *Proc. the 26th IEEE Symposium on Mass Storage Systems and Technology*, May 2010, Article No. 27.

[35] Amer A, Long D, Miller E, Paris J, Schwarz T. Design issues for a shingled write disk system. In *Proc. the 26th IEEE Symposium on Mass Storage Systems and Technology*, May 2010, Article No. 26.

[36] Jones S, Amer A, Miller E, Long D, Pitchumani R, Strong C. Classifying data to reduce long term data movement in shingled write disks. In *Proc. the 31st Symposium on Mass Storage Systems and Technologies*, May 2015, Article No. 12.

[37] Schwarz T, Amer A, Kroeger T, Miller E, Long D, Pâris J. RESAR: Reliable storage at exabyte scale. In *Proc. the 24th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, September 2016, pp.211-220.

[38] Pâris J, Schwarz T, Long D, Amer A. When MTTDLs are not good enough: Providing better estimates of disk array reliability. In *Proc. the 7th International Information and Telecommunication Technologies Symposium*, Dec. 2008.

[39] Amer A, Pâris J, Schwarz T, Ciotola V, Larkby-Lahet J. Outshining mirrors: MTTDL of fixed-order spiral layouts. In *Proc. the 4th International Workshop on Storage Network Architecture and Parallel I/Os*, September 2007, pp.11-16.

[40] Pâris J, Schwarz T, Amer A, Long D. Highly reliable two-dimensional RAID arrays for archival storage. In *Proc. the 31st IEEE International Performance Computing and Communications Conference*, December 2012, pp.324-331.

[41] Greenan K, Plank J, Wylie J. Mean time to meaningless: MTTDL, Markov models, and storage system reliability. In *Proc. the 2nd USENIX Workshop on Hot Topics in Storage and File Systems*, June 2010, Article No. 7.

[42] Yang M, Chang Y, Wu F, Kuo T, Du D. Virtual persistent cache: Remedy the long latency behavior of host-aware shingled magnetic recording drives. In *Proc. the 2017 IEEE/ACM International Conference on Computer-Aided Design*, November 2017, pp.17-24.

[43] He W, Du D. SMaRT: An approach to shingled magnetic recording translation. In *Proc. the 15th USENIX Conference on File and Storage Technologies*, February 2017, pp.121-134.

[44] Le Q, SathyanarayanaRaju K, Amer A, Holliday J. Workload impact on shingled write disks: All-writes can be alright. In *Proc. the 19th Annual IEEE/ACM International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, July 2011, pp.444-446.

[45] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage. In *Proc. the 6th USENIX Conference on File and Storage Technologies*, February 2008, pp.253-267.

[46] Patterson D, Gibson G, Katz R. A case for redundant arrays of inexpensive disks (RAID). In *Proc. the 1988 ACM SIGMOD International Conference on Management of Data*, June 1988, pp.109-116.

[47] Chaarawi S, Paris J, Amer A, Schwarz T, Long D. Using a shared storage class memory device to improve the reliability of RAID arrays. In *Proc. the 5th Petascale Data Storage Workshop*, 2010, pp.1-5.

[48] Le Q, Amer A, Holliday J. SMR disks for mass storage systems. In *Proc. the 23rd IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, October 2015, pp.228-231.

[49] Mesnier M, Chen F, Luo T, Akers J. Differentiated storage services. In *Proc. the 23rd ACM Symposium on Operating Systems Principles*, October 2011, pp.57-70.

[50] Pâris J, Estrada-Galinanes V, Amer A, Rincon C. Using entanglements to increase the reliability of two-dimensional square RAID arrays. In *Proc. the 36th IEEE International Performance Computing and Communications Conference*, December 2017, Article No. 27.

Quoc Le is a Ph.D. candidate in the Department of Computer Engineering at the Santa Clara University in Silicon Valley, focusing on disk performance and reliability in data centers. At the same time, he serves as a member of Technical Staff at Apstra, Inc., working on data center automation. Formerly, he was a senior software developer at data center business unit at Cisco where his team focuses on developing central management of Cisco Unified Computing System (UCS). He received his Master's degree in software engineering from the University of Wisconsin–La Crosse, in 2008.

Ahmed Amer is a faculty member of the Department of Computer Engineering, the Santa Clara University, Santa Clara. His research areas are systems, particularly storage and file systems, operating systems, and distributed systems. He is especially interested in alternative and upcoming storage technologies, and their most practical and effective application. He received his Ph.D. degree in computer science from the University of California, Santa Cruz, in 2002.

JoAnne Holliday is an associate professor of computer science and engineering at the Santa Clara University, Santa Clara. She also is a co-director of Center for Advanced Study and Practice of Information Assurance, Santa Clara University, Santa Clara. She is interested in doing research in distributed replicated databases, including concurrency control, distributed deadlock detection and avoidance, and wireless communication issues including reliable multicast and mobile, ad-hoc networks. She has recently begun research in wireless networks and distributed storage structures.