2-22-2019

Shingled Magnetic Recording disks for Mass Storage Systems

Quoc Minh Le

Santa Clara University

Department of Computer Science and Engineering

Date: Feb 22, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Quoc Minh Le

ENTITLED

Shingled Magnetic Recording disks for Mass Storage Systems

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor
Dr. Ahmed Amer

Thesis Reader
Dr. Nam Ling

Department Chair
Dr. Nam Ling

Thesis Reader
Dr. Silvia Figueira

Thesis Reader
Dr. Weijia Shang

Thesis Reader
Dr. Nicholas Tran

Shingled Magnetic Recording disks for Mass Storage Systems

By

Quoc Minh Le

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science and Engineering
in the School of Engineering at
Santa Clara University, 2019

Santa Clara, California

"Be kind whenever possible. It is always possible."
- Dalai Lama

Acknowledgments

This doctoral thesis would not have been possible without the help, support and patience of my advisor, Prof. Ahmed Amer, not to mention his advice and unsurpassed knowledge of storage systems. His willingness to give his time so generously, patience, motivation, enthusiasm, and immense knowledge has been very much appreciated. Also, I am lucky enough to have Prof. JoAnne Holliday as my former advisor. I am extremely grateful for her invaluable advice and support on both an academic and a personal level.

Besides my advisors, I would like to express my very great appreciation to the rest of my dissertation committee: Prof. Nam Ling, Prof. Silvia Figueira, Prof. Weijia Shang, and Prof. Nicholas Tran, for graciously agreed to serve in the committee and for insightful comments.

My sincere thanks also go to Dr. Katie Wilson and Ms. Heidi Williams, School of Engineering, for reviewing the paper, for valuable comments and feedback.

I wish to acknowledge the help provided by Prof. Darren Atkinson, Dr. Nirdosh Bhatnagar, Prof. Dan Lewis, Dr. Thomas Schwarz, Ms. Lisa Marie Jocewicz, and Dr. Christopher A. Kitts thorough my study.

I also thank the Department of Computer Engineering for their support and assistance since the start of my study, especially Ms. Apryl M. Roberts, Ms. Pam Lin, together with the other officers.

I thank my fellow postgraduate students in the Department of Computer Engineering at Santa Clara University: Kumar Raju, Yogesh Jhamb, Soohong Min, Rajesh Turlapati, Joseph Z. Issa, Johnny Wang, Samir Raizada, and Ming-tsun Hsieh for their stimulating discussions and for all the fun we have had during our times at SCU.

Last but not least, I would like to thank my parents, Loc Le and Nhung Luong, and my sister, Trang Le, for supporting me throughout my life. Finally, I would like to thank my wife, Han Chau, for her support and great patience.

Shingled Magnetic Recording disks for Mass Storage Systems

Quoc Minh Le

Department of Computer Science and Engineering
Santa Clara University
Santa Clara, California
2019

ABSTRACT

Disk drives have seen a dramatic increase in storage density over the last five decades, but to continue the growth seems difficult if not impossible because of physical limitations. One way to increase storage density is using a shingled magnetic recording (SMR) disk. Shingled writing is a promising technique that trades off the inability to update in-place for narrower tracks and thus a much higher data density. It is particularly appealing as it can be adopted while utilizing essentially the same physical recording mechanisms currently in use. Because of its manner of writing, an SMR disk would be unable to update a written track without overwriting neighboring tracks, potentially requiring the rewrite of all the tracks to the end of a "band" where the end of a band is an area left unwritten to allow for a non-overlapped final track. Random reads are still possible on such devices, but the handling of writes becomes particularly critical.

In this manuscript, we first look at a variety of potential workloads, drawn from real-world traces, and evaluate their impact on SMR disk models. Later, we evaluate the behavior of SMR disks when used in an array configuration or when faced with heavily interleaved workloads. Specifically, we demonstrate the dramatically different effects that different workloads can have upon the opposing approaches of remapping and restoring blocks, and how write-heavy workloads can (under the right conditions, and contrary to intuition) result in a performance advantage for an SMR disk.

We finally propose a novel use of SMR disks with RAID arrays, specifically building upon, and comparing to, 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 to the standard RAID 4, when using update in-place in RAID arrays, RAID 4SMR with garbage collection can allow not just the adoption of SMR disks with a reduced performance penalty, but offers a performance improvement of up to 56%.

Contents

List of Figures

List of Tables

Chapter 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 [60, 61, 27, 13]. Since an SMR 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 SMR 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 input/output (I/O) workload presented to the device.

SMR drives could behave differently for write and re-write operations. Write operations when come to SMR drives will be written sequentially to an arbitrary band. Seagate, one of the most major drive suppliers, published an article [57] 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 article [62] enables the customers to help themselves, states that all physical sectors are written sequentially in a direction radially and are only rewritten after a wrap-around. The write behavior of an SMR drive is also confirmed by A. Aghayev et al. [1] in an effort that combines software and hardware techniques to discover key properties of drive-managed SMR drives.

As of 2018, there are many cloud storage providers such as Google Cloud Storage [12], Dropbox [9], BackBlaze [4], *etc.* 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. These use cases should be perfect for SMR disks. 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 SMR disks. We proposed a scheme for using SMR disks in RAID arrays, named RAID 4SMR. We also evaluated our RAID 4SMR scheme with garbage collection and analyzed the reliability of the scheme.

In the remainder of the manuscript, we first present how the technology behind shingled writing changes the functional behavior of a disk drive, and the data layout approaches that can be used to address these changes. We then present an initial assessment of a varied set of I/O workloads, as ultimately the effectiveness of any approach appears to be more dependent on the nature of the workload than the specific parameters of the shingled writing mechanism. Of particular interest is our finding that changes to device blocks are very heavily concentrated in hot zones, and that most blocks are

written only once over extended periods of time [2], and that the prevalence of updates (as opposed to one-time writes) is one of the most important factors determining the best approach to data layout and management. Most surprisingly, we demonstrate how attempts to remap blocks to achieve sequential write behavior can, at one end of the spectrum be very detrimental to performance. On the other hand, a predominantly write-heavy workload can actually result in a performance gain for a shingled disk over a regular disk. This is contrary to the intuition that shingled-write disk performance would suffer due to the restrictions it faces on updating data in-place.

Later, 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. 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 changed the name from RAID 4S to RAID 4SMR to avoid confusion with RAID 4S by Rosie *et al.* [52]) which explore how SMR disks can be combined with conventional magnetic recording (CMR) or standard hard disk drive (HDD) disks to achieve an efficient and reliable RAID array. 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 allow not just the adaption of SMR disks with a reduced performance penalty, but offers a performance improvement of up to 56%.

Chapter 2

Related Work

Rosie *et al.* [52] use the name of RAID 4S for adding faster solid-state drive (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 high number of update-in-place operations [30].

Jin and Liu *et al.* [19, 35] proposes an SMR RAID file system. The difference is, 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 *et al.* [37] tried to employ SMR RAID with SSD. Aghayev *et al.* [1] 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 [60, 61, 27, 13, 22, 24, 25, 55, 63, 11]. Most proposed techniques revolve around some form of log-structuring of the data [21, 48, 65, 34, 32, 16, 15]. The log-structuring mechanisms owe their designs to the original log-structured file systems [43, 50, 51], and subsequent works applying the same techniques in areas as diverse as databases, tertiary storage, and tape-based

filesystems [23, 59, 8, 10, 10, 23, 33, 36, 39].

Recent works describing the management of data on SMR disks have been described by Gibson *et al.* [11], and by Casutto *et al.* [5]. 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.* [2, 20] explored a spectrum of design parameters for SMR 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) [56, 46, 3, 45, 14].

Ming-Chang *et al.* [64] presents a virtual persistent cache to remedy the long latency behavior of host-aware SMR. Weiping *et al.* [17] introduces an approach to SMR translation which adapts a drive-managed SMR data management scheme.

Chapter 3

Background

3.1 Shingled Magnetic Recording

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.* [55] to describe the physical limit that hard drives are rapidly approaching. In essence, while increasing storage density results in physically smaller bits being recorded on the medium, such a reduction in physical scale renders the written data more unstable at room temperatures. This is the super-paramagnetic limit described by Charap *et al.* [7], and while this limit has been exceeded over the past decade, primarily thanks to the development of GMR heads, AFM media, and orthogonal recording, the fundamental problem of a recorded bit being too unstable to reliably hold data is nonetheless fast approaching. Increasing the stability of the recording medium allows a reduction in the minimum physical size of a unit of recorded data, but also requires that a more powerful magnetic field be applied. This is because a more "stable" medium is a less writable medium. But using a more powerful magnetic field, means that a larger area is affected by that field, defeating any density gains we hoped to achieve. So we are caught between the required writability of the medium, our ability to write physically smaller areas, and our ability to successfully read back such data. This is the media trilemma illustrated in Figure 3.1.

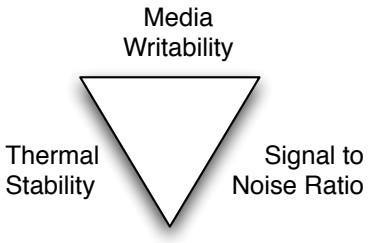


Figure 3.1: The media trilemma (a term coined by Sann *et al.* [55])

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. Some alternative approaches to shingled magnetic recording have included patterned media, which adds considerable complexity to media production, or some variant of assisted magnetic recording. The former, patterned media, requires the isolation of nano-structures on the media surface with the intention of limiting the negative impact of recorded bits on their neighbors. This is a radical departure from the relative simplicity afforded by current media and would result in a more complex production process, limited by our ability to fabricate disk platters with adequately small structures. The second alternative requires assistance from microwaves or lasers, serving as directed heat sources. Such a heat source would be directed at a narrow area of a more stable medium (one with lower writability), rendering that focused area temporarily more writable. Such microwave-assisted magnetic recording (MAMR) [66] or heat-assisted magnetic recording (HAMR) [26, 38, 53] attempts to overcome the media trilemma by using a heat source that can be narrowly focused to artificially limit the impact of a more powerful magnetic field to the area upon which we've focused the heat source. Naturally, any such approach requires a more complex read-write head assembly, featuring a laser in addition to the magnetic read-write component. Shingled magnetic recording, on the other hand, avoids any such added complexity to the medium or the head assembly, it does, however, require minor

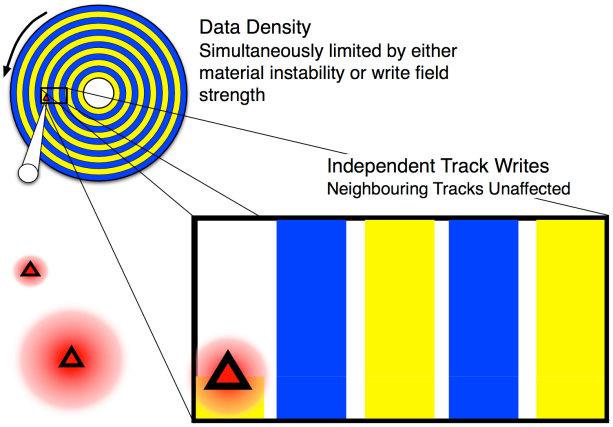
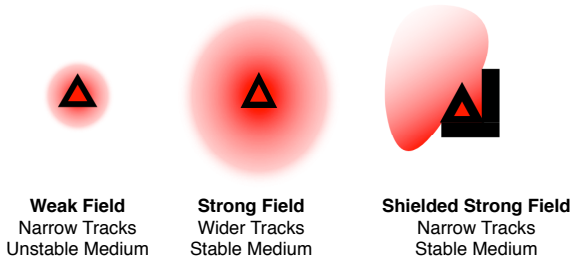


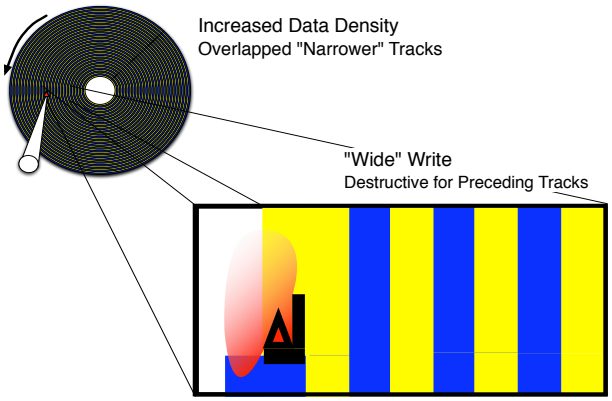
Figure 3.2: A conceptual view of tracks, as written with a current disk head. Decreasing media writability (to improve stability) for the sake of increasing density would have an adverse effect on track density if the disk head resulted in "wider" tracks.

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 Figure 3.2 we see how data is organized on a disk in a series of adjacent "tracks" each of which is distinctly written and distinctly 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) are at 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 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 (so 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.



(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 an SMR disk gains storage density by overlapping successively written tracks, leaving "narrower" tracks in its wake.

Figure 3.3: Shingled magnetic recording, increasing data density through the use of overlapping tracks, written through the use of a shielded disk head.

Specifically, a magnetic shield is added to the trailing sides of the head, as shown in Figure 3.3(a). In this manner, as a track is written for the first time, it will be written as a "wide" track. However, such width would only impact the current track and 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 to effectively overlap them. Resulting in a shingled track arrangement, where all that is left of a track 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 Figure 3.3(b). However this increased density comes at the expense of rendering any subsequent attempt to update these narrower preceding tracks destructive.

3.2 Organizing Tracks into Bands

Our prior work, as well as that of other researchers has explored techniques to alleviate the effects of this restriction, typically through some form of log-structuring of writes to defer the need to update data in-place [2, 5, 11]. Casutto *et al.* [5] 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.* [2] explored a spectrum of design parameters for SMR disks, including alternative interfaces such as object-based stores, or file system-based approaches to addressing the new disk behavior.

A disk for which all the tracks are overlapped would be impractical for use as a random-access block storage device, and so a suitable layout scheme for the written blocks and tracks is essential to maintain existing functionality of magnetic hard drives. If a SMR disk 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. Attempting to overwrite a block in the last track could be accomplished without harm, but attempting to update any previously written track would result in the overwriting of an adjacent track for which no update request has been made (and the contents of which we may have not recently read). 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 the track being updated, but as each track would itself have to be written back to 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.

When log-structuring is not possible, but we still want to avoid the need to update

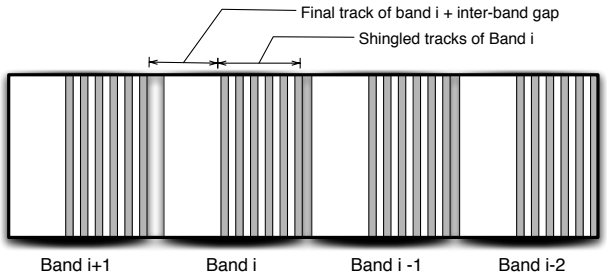


Figure 3.4: Logical view of a SMR disk divided into bands, allowing the in-place update of a band, athough at the expense of a destructive track write within an individual band.

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 SMR disk can be arranged into distinct bands [22, 11, 2, 5]. We illustrate the logical view of such bands in Figure 3.4. By limiting the number of overlapping tracks in a band, we create a disk that allows the random update of the last track in a band, and at worst requires only the rewriting of all the tracks within a band if there is an update to one of the overlapped tracks within the band. Thus, we want to use larger bands to increase the storage density of the drive, but we also want to limit the size of the bands to avoid the penalties in extra activity and delays that would be caused by update and move activity. The number of tracks assigned to a band does affect the performance of a shingled disk, but a consideration that's at least as important is the workload observed and the manner in which it is handled by the data layout scheme.

3.3 Data layout for SMR

To use a SMR disk as a regular disk requires one of two basic strategies at the block level: updating blocks in-place and performing all copies needed to make sure adjacent tracks are preserved; or remapping a block or track number so as to relocate it physically to the end of a band with free space. The former approach, implementing update in-place, avoids block remapping, which might disturb any locality being attempted by

the overlying software. However, in-place updates have the disadvantage of requiring additional operations, with the associated performance costs. Block remapping, on the other hand, is in essence a copy-on-write strategy similar to that employed by log-structured filesystems [43, 51, 50]. In fact, schemes based around mapping log-structured file system data structures to shingled disks have been proposed as a strategy for shingled disks [22, 5]. Prior efforts have shown how such data structures could be used to implement file-systems on tape-based systems [23, 65, 48]. We show that the success of this strategy for SMR disks is heavily dependent on the nature of the workload.

While it is tempting to think of a SMR disk as a sequential write device, similar to tape, an important distinction is that a SMR disk remains as adept at random read operations as a regular disk. This makes it very different from a sequential-write and *sequential-read* tape system. When a workload results in a large number of random reads, it might be important to preserve the locality expected by the overlying software; in other words, to guarantee that adjacently numbered blocks are physically near each other. While such an assurance could be offered by avoiding the remapping of blocks, or by actively attempting to relocate blocks and tracks so as to restore their logical adjacency, the success of any such activity (in fact its very necessity) is dependent on the workload. While precise performance of any strategy on any future disk will depend on the physical characteristics of the device, we aim to provide a model to objectively evaluate logging and in-place update strategies. Examples of physical characteristics that may dramatically impact final performance include whether or not a shingled disk will employ TDMR recording [24, 25, 55, 63, 11] which would result in slower read operations due to the need to await multiple rotations. To allow for a more objective evaluation of the impact of workloads on potential layout strategies we introduce in the System Model section a logical distance-based view of disk activity and four comparative models for our evaluation purposes. These models are explained in section 4.

In a later section we will propose RAID 4SMR which is based on RAID 4 [47] the industry standard solution for block-level (not chunk) striping with a dedicated parity disk which allows parallel read/write.

Chapter 4

System Model

In this section, we introduce the model and metrics used to evaluate our proposed method of handling SMR disks.

The most notable characteristic of an SMR disk will be its inability to perform an in-place update of any tracks that have been previously over-written. To accommodate the use of SMR disks as regular hard drives will require remapping or dynamic relocation of data blocks that are need to be rewritten, and to this end we are faced with a spectrum of solutions. At one end of this spectrum is a continuous logging model that employs copy-on-write to relocate all written blocks, thereby never attempting to perform an update-in-place. At the other end of the spectrum is attempting to perform an update in-place, and avoiding the destruction of previously written data by re-writing any affected tracks. Assuming no memory restrictions, it would be possible to achieve the latter solution by reading and buffering all affected tracks (up until the end of the band) and then writing back the newly updated track and those tracks we have just buffered. Unfortunately, this is not particularly realistic, and so we would need to consider the effects of limited buffer capacities on the system. We introduce four system models:

1. a standard disk, that is free of the destructive-write limitations of a shingled disk

2. a purely logging disk, that makes no attempt to update modified data, but employs a pure copy-on-write approach writing new versions of updated blocks at the next

free location

3. a disk that attempts to update in-place, but preserves any adjacent tracks with the benefit of unlimited buffering capacity

4. a disk that attempts to update in-place, but is restricted to a fixed buffer capacity for use in preserving the remaining data in the band

None of these approaches are perfect representations of an ideal solution, but they allow us to investigate the impact of basic elements of the various approaches. Comparing the effects of the purely logging model against the update-in-place models allows us to gauge whether or not it would be important for a specific workload to avoid the inevitable fragmentation of data, or whether it would be beneficial to optimize writes by updating the disk as a log. Comparing both update-in-place schemes allows us to gauge the importance of a large memory buffer, or more broadly, the impact (and necessity) of employing substantial non-volatile random-access memory (NVRAM) buffers. Such buffers might be useful to absorb the bulk of random updates, allowing an SMR disk to deal with the more sequential and stable workload that would result. However, as we will demonstrate, for some workloads, the negative impact of implementing an update-in-place strategy for a shingled disk can be almost completely masked using relatively small buffers.

Simulation Model

To evaluate the impact of varied workloads on an SMR drive, we attempted to build flexible models, and 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 logical block addressing (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, independent to which model the disk uses, 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. These metric are related to time as well. More activities take more time and space. Blocks activity is proportional to the time of operations, but it is not specific because time very depends on a specific configuration, but this "block movements" gives us a metric of universal. The number of block movements is the difference between the logical address of the first block visited and the next block visited. We have 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 1 logical track direction change. We have 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.

The direction changes of track and band depend on how big the track and band size are. In the simulation, we assume the track and band size are from 200 to 1400 (increment 600 for each step). That means the smallest band size is 200 (blocks per track) x 200 (tracks per band) x 4 KB (4 KB = size of disk block, current standard) =

160,000 KB = 160 MB, the biggest band size is 1,400 x 1,400 x 4 KB = 7,840,000 KB = 8 GB.

We created a simulator to model the behavior of an SMR disk with four specific schemes, matching the four models. They are as follows:

1. **Scheme 0**: Standard disk (baseline)

   This represents a standard disk with the ability to perform random in-place writes without risk of destroying any adjacent tracks. We included this scheme in our simulation as a baseline representative of current disk technology.

2. **Scheme 1**: SMR disk as logging disk

   This is also a baseline scheme of sorts, as it avoids the impact of shingling by assuming that the disk has enough capacity to absorb all writes in the form of a continuous log, always creating a new copy of any data that needs to be updated at the end of the disk. Such a model is effectively write-optimized.

3. **Scheme 2**: SMR disk with the ability to update in-place, with **unlimited memory (buffer)** to hold data being relocated (existing data blocks in the band)

   This is one of the smart schemes with unlimited (or ideal) buffer memory capacity available. In this scheme, whenever we want to update a block, the rest of the band needs to be rewritten to preserve the data. Thanks to the guaranteed ample buffering (unlimited buffer), this approach can complete an update of a track in one read-update-write sequence.

4. **Scheme 3**: SMR disk with the ability to update in-place, with **limited memory (buffer)** to hold data being relocated

   For this scheme, we assume a limited transfer buffer available to support attempts to update data in-place. The smallest buffer size is 64 MB based on the realistic

sizes of current buffers in commercial disks available today, and the largest buffer size we evaluated was 8 GB. We have found the memory size to have great impact on the logical performance of write operations, but the memory capacity needed to effect such a great impact varies dramatically based on the observed workload. Since an SMR disk cannot update-in-place (write-in-place) because the data next to the block is affected, whenever we want to update a block, the rest of the band needs to move to a safe place before moving back to the original band. In case buffer capacity is less than band size, we will fill up the buffer with data from the blocks next to the updating block, empty it to the next available block at the end of the disk, fill up the buffer again, empty it again, etc., until the last block of the band is moved. The same set of operations is done again when we have finished updating the block and moving the rest of the band back to the original band.

Chapter 5

Workload Evaluation

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 [31]. To evaluate the impact of shingled writing under varying workloads and for different device parameters (band size and buffer capacity), we conducted simulations of the behavior of our four system models against a wide variety of workloads. While it might be tempting to simply classify workloads according to their ratios of read-to-write operations, it is actually more important to consider the nature of the writes, and to that end we start by characterizing the different workloads to distinguish repeated updates from one-time writes.

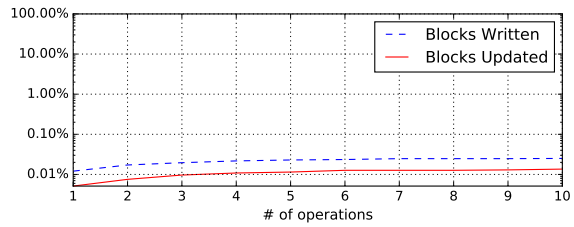
5.1 Workload Characterization

We have evaluated the Shingled Magnetic Recording, SMR, models against a variety of workloads. The workload types collected are Block I/O level traces [41] drawn from a variety of system types, block traces reconstructed from web server HTTP request logs [42], 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 have been 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

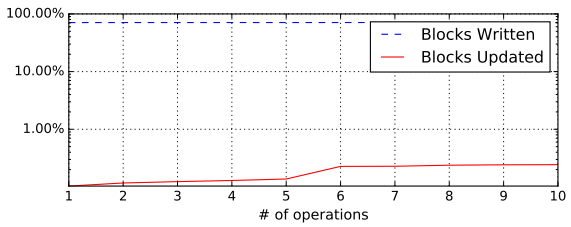
of our file system traces was drawn from a file system being used to host the image files of a local VMWARE installation. These workloads were collected from a wide pool of systems, which varied greatly in the total number of operations observed, and in the mix of reads, writes and updates. We define 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 showed a variation of the percentage of updates across and within workload types. Percentages of observed read operations varied from 0.02 to 99.78%, while writes varied from 0.22% to 99.98% and updates were seen to range from 0.001% to 99.49% of the total number of blocks accessed.

A total of 35 different workloads were evaluated and tested against our shingled disk models, and of particular note was the impact of varying percentages of write and update operations. We highlight four examples of such varied mixtures in Figure 5.1. Specifically, a workload with very low writes (0.03%) and very low updates (0.02%) as shown in Figure 5.1(a), one with medium writes (71.04%) and low updates (0.25%) as shown in Figure 5.1(b), one with high writes (98.96%) and low updates (9.55%) as shown in Figure 5.1(c) and finally one with very high writes (99.98%) and low updates (4.57%) as shown in Figure 5.1(d).

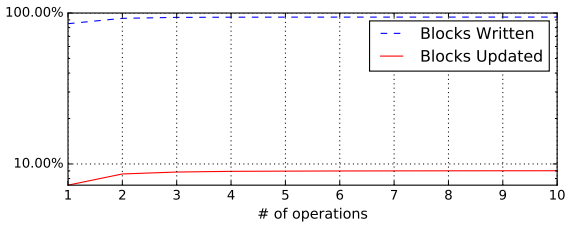
As an SMR disk cannot simply perform an update in-place, we paid particular attention to block update operations. It is these operations which, unless they happen to fall at the last track of a band, would require us to address the update-in-place restriction. Our analysis differs from prior art [2] 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. For example, in Figure 5.1 we plot the percentage of blocks



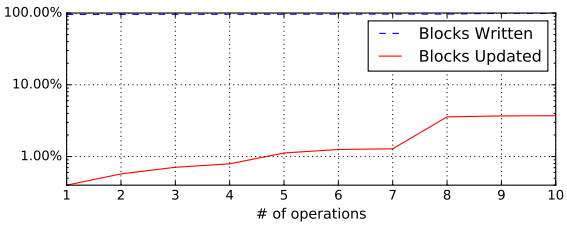
(a) NASA: 0.03% Writes, 0.02% Updates



(b) WEB3: 71.04% Writes, 0.25% Updates



(c) FS-VMWARE: 98.96% Writes, 9.55% Updates



(d) RSRCH1: 99.98% Writes, 4.57% Updates

Figure 5.1: Four example workloads highlighting the extreme variability of write operations, and also illustrating the variety in update percent (and its lack of necessary correlation with the percentage of writes)

that were observed to be writes or updates. At each data point, we plot the percentage of all blocks that were written 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. For example, the WEB3 and RSRCH1 workloads (in Figures 5.1(b) and 5.1(d)) demonstrate how these values can diverge. For the RSRCH1 workload, we see that almost all blocks written were written at most once, with hardly any written more than once. For the WEB3 workload, we see similar behavior, but with a smaller percentage of blocks experiencing multiple writes. In both workloads, the percentage of updates was much lower. The reason for the disparity lies in the fact that blocks written once contribute to the large disparity between the two series. When results varied, the variation tended to be increasingly dramatic at scale, and so most of our results are

presented on a log-scale.

As we can see from Figure 5.1, there is a tremendous variation among traces in both the percentage of writes and the percentage of blocks that are updated infrequently as well as frequently. Results that show a marked increase in block percentages (the y-axis) as the maximum number of operations is raised (increasing values along the x-axis) are indicative of a workload that experiences frequent operations to the same blocks, whereas largely flat series indicate that the majority of blocks experience the indicated operation at most once. We have found the percent of writes and updates to be highly indicative of the relative performance of the system models we have evaluated.

5.2 System Model Performance Results

The choice of band size, *i.e.* the number of shingled tracks between each inter-band gap that allows us to avoid impacting neighboring tracks, has a direct influence on the performance of an SMR disk relative to a standard disk. A normal disk is logically equivalent to an SMR disk with a band size of one track. And so we compare the performance of the four models for a range of band sizes, starting with a modest 160 MB band size, and reaching up to just under 8 GB (7.84 GB). But we have found that the update behavior observed in the workload has the greatest impact on the shingled write models. The greater the update percent, the greater the positive impact of logging (block relocation to the end of a running log) and the greater the negative impact of attempting to restore updated data to original locations (by effectively implementing an update in-place atop an SMR disk).

Figure 5.2 shows the write and update behavior of WDEV2 [41], the workload observed to have the highest update percent, while the lowest update percent was observed with the NASA [42] workload described in Figure 5.1(a). In Figure 5.3, we show the number of logical block movements that result under the four system models: a standard

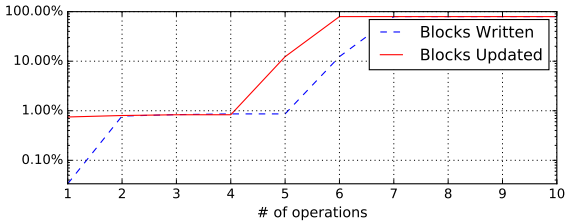


Figure 5.2: The WDEV2 workload, demonstrating the highest percentage of updates, which were also particularly notable in that updates to individual blocks tended to be frequently repeated

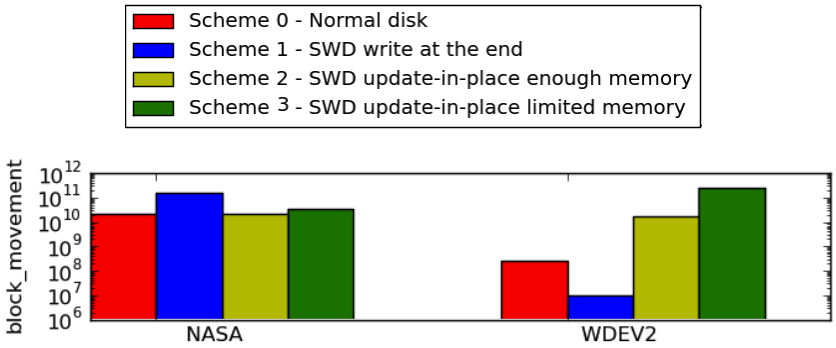


Figure 5.3: Comparing the logical block movements resulting from all four disk models, between the NASA (lowest write and update percents) and WDEV2 (highest update percent) workloads

disk, a logging disk, an in-place update implementation with unrestricted buffer space, and the average behavior of the corresponding schemes with limited buffer capacities. The greater the value of "block movement," the more logical distances that would need to be traversed (hence "movement"). While this graph shows logical block movements, the results appear correlated with track movements, which are directly related to the amount of physical activity (and corresponding latencies) incurred by the disk head. As we mentioned earlier, this "block movements" related to total activities done, therefore it is proportional to time but independent to a specific disk.

The first data bar in each cluster in Figure 5.3 shows the logical movements of the standard disk model. The differences between the two workloads are a result of their different lengths and behavior. The second data bar for each set represents the purely logging append-only SMR disk model, and as we can see, this shows increased movement

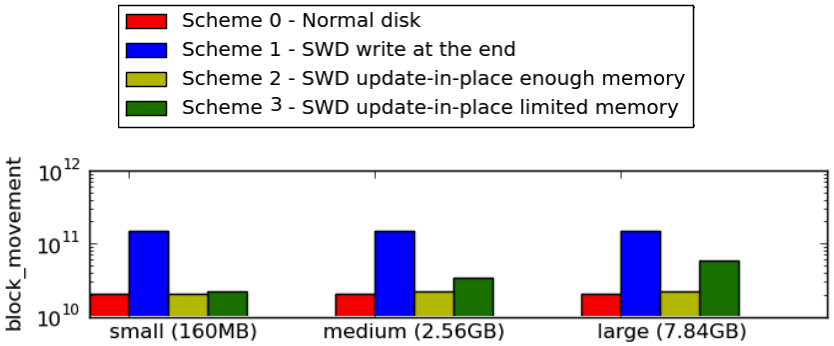
(poorer performance) for the NASA workload, which was predominantly a read-heavy workload, while the write (and update) heavy WDEV2 workload experiences a dramatic improvement in performance. This is the result of the block-relocation allowed by logging and its subsequent write-optimized behavior. When attempting to deal with updates in-place by updating the band in which the updates need to occur, we see the penalty of increased movement in the third and fourth data bars. This degradation is not notable for the NASA workload, arguably a best-case scenario for SMR disks attempting to mimic regular disk behavior (without the benefit of the intensive block remapping required for a logging scheme). The relative impact on the update-heavy WDEV2 workload is dramatically worse. As would be expected, operating with a limited buffer for band updates leads to increased activity in order to perform the update in multiple steps. How many steps, and the degree of this impact depends on the size of each band.

5.3 Impact of SMR parameter on behavior of SMR

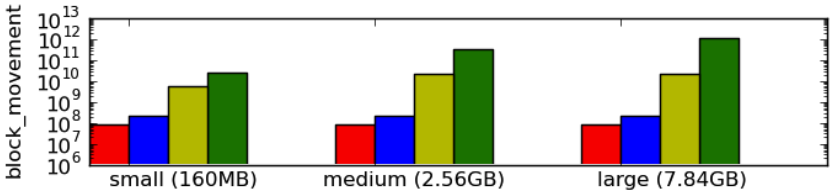
To illustrate the impact of band size, we consider the impact of the four workloads discussed in Section 5.1 across varying band sizes and under the four system models. As we saw in Figure 5.3, the predominantly read-heavy NASA workload exhibits the least degradation from utilizing shingled writing. Nonetheless, we see an increasing negative impact correlated with an increase in the band size, most consistent when moving from a limited to an unlimited buffer capacity. This is to be expected, as the size of a band is an upper limit on the amount of data that is affected by an update in-place. A bigger band size would demand a bigger buffer be utilized when a band needs to be updated, to avoid moving other blocks to a temporary or new location. This increasing impact of band size is consistent throughout all traces evaluated, but its impact is surprisingly muted when compared to the impact of the update and write rates of the workload.

The NASA workload, while having the lowest update and write rates among all 35 workloads considered, suffers considerably when a purely logging disk model is employed. In fact, WEB3 and NASA, both workloads drawn from systems that run web workloads with their tendency to be reference-heavy and update-infrequent, show a penalty for attempting to remap blocks (even with an idealized model such as ours). It is important to note that the WEB3 workload had a majority of its operations manifest as write operations (typical when the majority of read operations are absorbed by effective caching strategies). This highlights the importance of considering update rates, and not simply read and write ratios, as an SMR disk suffers no adverse effects when data is written only once. The negative impact of the logging disk model (the second model) upon the NASA and WEB3 workloads is contrasted with its positive impact on the FS-VMWARE and RSRCH1 workloads. In these workloads we see the vast majority of operations are writes, and a significant (but still a minority) of operations are updates. For both these workloads, we see a reduction in movement on the order of 50× to 150×. Similarly, the positive impact of the logging disk model on the WDEV2 workload is because of an extremely high percentage of updates and writes. The head in this case must almost move to the beginning of the available blocks and add up data without moving back and forth to move data. This illustrates the tremendous potential for block remapping strategies on SMR disks in the presence of heavy write and update workloads. The greatest improvement was observed for the RSRCH1 workload, which also featured the largest percentage of writes. The negative impact of attempting to perform updates in-place is also clearly demonstrated.

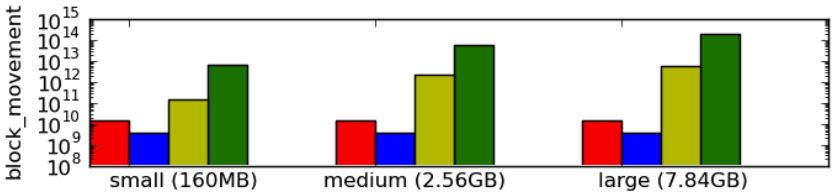
To absorb the impact of a write-in-place policy (which can be beneficial for heavily read-biased workloads), it is important to select an adequate buffer capacity. This is particularly true when we are dealing with volumes of data on the order of several GBs, as it will require a memory buffer that is both fast and non-volatile. Flash or alternative



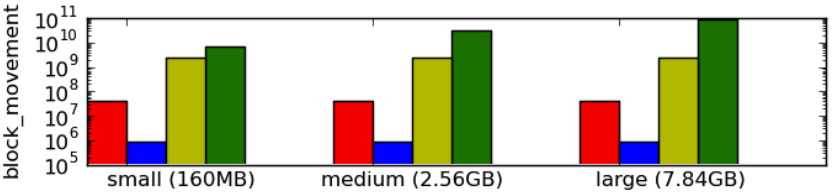
(a) NASA varied band capacities



(b) WEB3 varied band capacities



(c) FS-VMWARE varied band capacities



(d) RSRCH1 varied band capacities

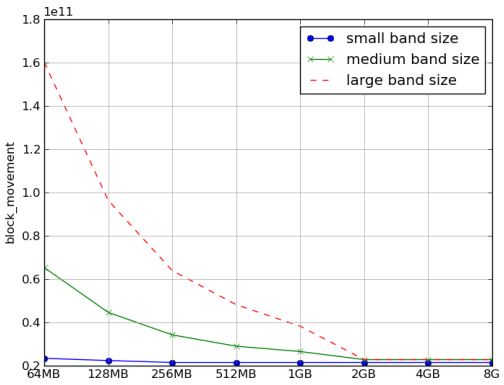
Figure 5.4: The impact of band size on logical movements for the four disk management models

storage-class memories could be employed to this end. Our workload evaluation has shown that the necessary memory sizes to mask any negative performance impact due to insufficient buffer capacity can vary considerably. Surprisingly enough, we found that the amount of memory buffering needed appears to converge rapidly to a consistent value across multiple band sizes. To this end, we expand the fourth system model results (64 MB buffer capicity) presented in Figure 5.4 to illustrate the block movement behavior under different memory capacity limits. As before, higher values are indicative of poorer performance, but we dispense with the logarithmic scales in these graphs.

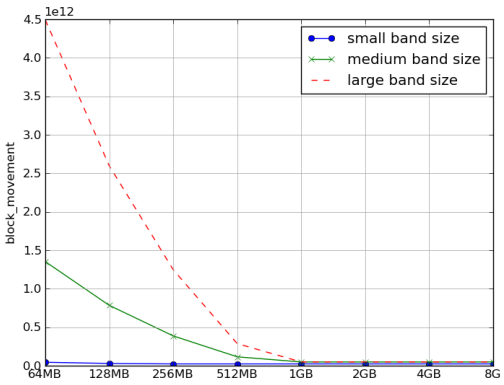
In Figure 5.5 we see that for all three band sizes, providing a buffer anywhere from 128 MB to 8 GB is sufficient to counteract the negative impact of larger band sizes. The fact that this capacity can be as low as 128 MB for one of the most problematic workloads (RSRCH1) is particularly noteworthy, as this is practically an all-write workload, and yet it appears to do all right when approached with logging strategies or minimal additional buffer capacity.

5.4 SMR in Server Environments

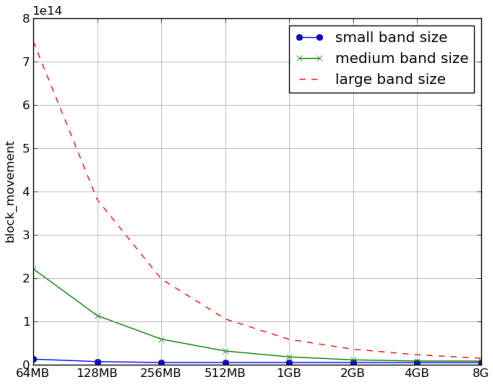
In chapter 6 "Building SMR-Aware Disk Array", we propose one possible approach to building an SMR aware disk array suitable for use in a server environment. We, therefore, first evaluate the impact of data placement and benefits or lack of them from interleaving workloads that involve writes originating from multiple sources on these disk arrays. Whether an array of SMR 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 an SMR drive. We found that a workload that originates from a heavily interleaved mix of sources is detrimental to an SMR disk performance. We reached these preliminary conclusions through the replay of recorded workload traces. To evaluate the impact



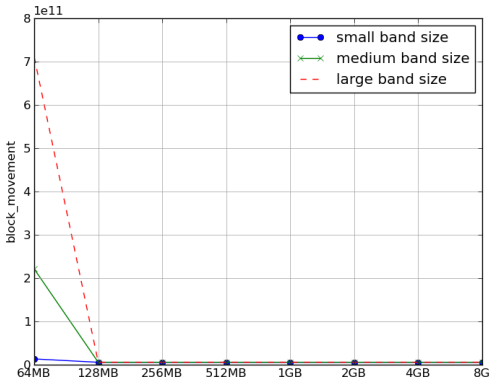
(a) NASA



(b) WEB3



(c) FS-VMWARE



(d) RSRCH1

Figure 5.5: The impact of buffer size on logical movements for the fourth (memory-restricted replacement) disk management model.

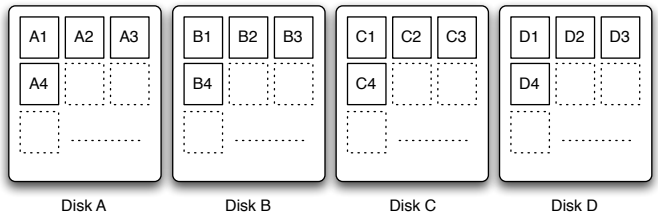


Figure 5.6: Logical view of a simple array of disks. In the **striped** arrangement, blocks 0, 1, and 2 are arranged as A1, B1, and C1. In **pure** arrangements, blocks 0, 1, and 2 are arranged as A1, A2, and A3.

of shingled writing when employed on disks arranged in array, we evaluated several 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 do not consider disk parallelism in this case to simplify the results.

In section 5.2 "System Model Performance Results", we have evaluated the performance of a single SMR disk against a variety of workloads. The workload types collected were Block I/O level traces [41] drawn from a variety of system types, block traces reconstructed from web server HTTP request logs [42], and new block-level traces which we collected from general file system usage over several months. From workload traces, we have also been able to generate workloads representative of specialized applications. For example, one of our traces was drawn from a filesystem being used to host the image files of a local VMWARE installation, while others were reconstructed web server workloads. These workloads were part of a larger collection we compiled from a pool of 35 different real-world workloads. These workloads varied greatly in the total number of operations observed, and in the mix of reads, writes and updates, and from them we extracted four disparate workloads as representatives for use in the current experiments.

Figure 5.6 shows the logical arrangement of blocks we evaluated, while Figure 5.7 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 Figure 5.7 were based on an SMR disk

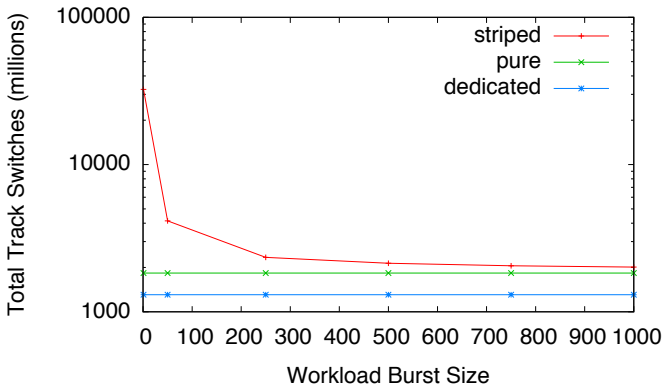


Figure 5.7: Disk activity when replaying multi-source traces against a simulated array of SMR disks.

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 which 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 one (where the interleaving is maximized) up to bursts of a thousand operations. Finally, the **dedicated** results represent the behavior of the SMR disks when each disk is dedicated to an individual source workload.

Figure 5.7 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 forty 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 6, "Building SMR-Aware Disk Arrays", we will propose RAID 4SMR which is a disk array system based on RAID 4 [47] using SMR disks.

5.5 Offering a Modified Object/File Interface

Using shingled disks with applications that do not require any data updates is simple, and requires no modification of device firmware or drivers to accommodate the shingled nature of the disks. A continuous video recording application, for example, recording on a continuous loop, would be able to write to an underlying storage system composed of shingled disks. Those disks could write the video stream data to consecutive tracks, never overwriting the preceding tracks, but simply looping to the first written track on

the first disk. At this point, the tracks that will be destroyed by an overwrite would be the tracks holding the oldest video data that is due for erasure. In such a scenario, simply using the SMR disks as-is would be the most efficient approach to construct such a system.

With a file system or object store, the system selects specific blocks to hold the data that is contained within individual objects or files. Because of this, such an implementation has considerable freedom regarding how the data is placed. Since it need only guarantee that it will provide the same contents in response to requests for a specific object or file, such a system may freely relocate and reassign individual portions of such objects to new blocks as needed. Simultaneously such a system would not need more information than the metadata it is already maintaining for the purposes of its allocation policy. When general storage systems implement copy-on-write or log-structure layout policies, such as that employed by write-anywhere file layout (WAFL) [18] or ext3-COW [49], then write-in-place updates are largely unnecessary. In the case of WAFL, this allowed the system to utilize an inexpensive and simple RAID4 arrangement of data on its underlying disk, thereby offering a tremendous cost advantage over competing technologies. This sort of arrangement would also be amenable to shingled disk applications.

It is possible to use SMR disks without modifying their interface by requiring the file or object system to be aware of the destructive nature of updates to tracks within a band and to ensure than any data at risk of being overwritten has been moved prior to writing. This can be achieved by writing data updates to new locations, in a copy-on-write (COW) fashion, thereby invalidating the older versions of the data (allowing them to be safely overwritten in their original locations). Such an interface can be provided by the device itself (as in the case of the Seagate Kinetic drives which act as network-attached key-value stores), or by a layer of the software stack, such as that utilized

by filesystems like CEPH or the SWIFT/Openstack architectures, or by a complete overlying fiesystem adapted for the shingled nature of the underlying disks.

Adapting an overlying filesystem to allow for the freedom to write "anywhere" was demonstrated in the WAFL filesystem developed by NetApp, and which formed the basis of their first products [18]. By using a copy-on-write scheme for both the files' data blocks, and for the metadata that described those files, the system essentially rendered the underlying storage system into a target for streaming block writes that could be grouped together into convenient chunks of data that coincided with the size of a RAID [47] stripe. This meant that all underlying block write operations could be guaranteed to require no updates to the parity disk (as a complete stripe is being written, this means that the data being written includes a completely new value for the parity of that stripe, and the blocks could overwrite the existing RAID stripe in one efficient parallel write operation). An interesting side effect of this approach is that NetApp was able to utilize relatively inexpensive RAID 4 device arrays (as opposed to RAID 5 systems that attempted to alleviate parity update load by automatically varying the device assigned to hold the parity block). This contributed significantly to the competitiveness of NetApps early storage offerings, as it meant that the modified file system had allowed improved system performance and reduced the cost of the hardware required to realize that improved performance.

In order for such a scheme to be feasible, it was necessary to hold newly written data in a non-volatile memory until it could be guaranteed that an entire stripe could be written out to the disk devices. A similar approach is possible for SMR systems, but the overhead of this strategy (simply using nonvolatile memory to hold a large pending write buffer) may be unnecessary for individual devices, and considerably more expensive for disk arrays. The problem of cost with disk arrays is that we would need to buffer more than a simple parity stripe (a number of blocks no larger than the total

number of devices in an array), but would in fact need to buffer a stripe composed of entire bands in a stripe. So instead of a collection of x blocks, it would be a collection of $x \times y \times z$ blocks, where x is the number of devices, y is the number of blocks in a track, and z is the number of tracks in a band. This is necessary to guarantee that data updates do not destroy data when used with a straightforward RAID implementation, which we will now review. We will present one possible, yet novel, alternative approach to offering a block interface for disk arrays based around SMR disks.

Chapter 6

Building SMR-Aware Disk Arrays

In computer storage, there are several different standard and non-standard redundant array of independent disks (RAID) configurations such as RAID 0, 1, 2, 3, 4, 5, 6, 10, etc. 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 chose 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 create more overhead compared to RAID 4.

Shingled write disk with SMR technology can help us triple data density in the future [61], 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 [30]. We now propose RAID 4SMR in Figure 6.1 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

a 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).

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 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 the SMR is most suitable for archival storage or a write one read many (WORM) disk, we have designed our SMR RAID to be a solution for reliable data storage arrays. The reason we chose RAID 4 instead of other RAIDs is that RAID 4 stores frequently updated parity blocks on a dedicated disk, left alone 3 data disks can be all replaced by SMR disks.

One of the advantages of the proposed 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 Figure 6.3. The Data HDD does not need to be the same size as 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.

We anticipate a volume size for this simple standalone RAID 4SMR would be around 30-40 TB. This could be accomplished with an array of 3 x 10 TB SMR disks and 2 x

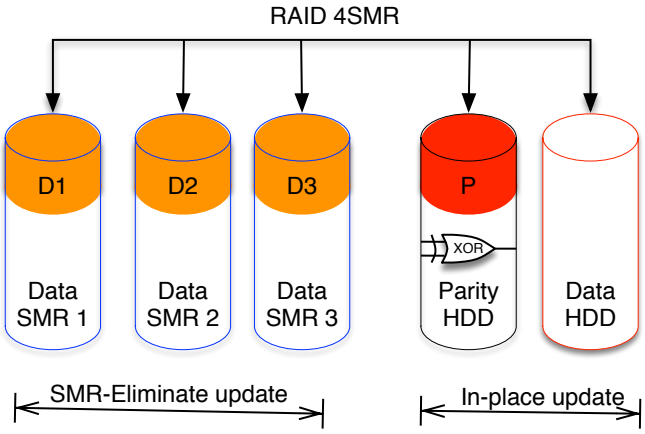


Figure 6.1: When the data first write to the array, parity disk will store P = XOR(D1, D2, D3, DataHDD). Data HDD blocks are expected to be zero-initialized.

10 TB regular HDD disks (for both parity and data disks).

In Figure 6.1 we illustrate our approach to maintaining a block interface for a disk array built around SMR devices. In this example, data is held primarily on disks D1 through D3, which are all SMR devices, while parity is held on disk P, which is not an SMR 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 to D3) only updated 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 become the potential performance bottleneck as well as the Parity HDD disk. We recommend a high-performance disk be used in this case.

Since the Data HDD can be updated in an update in-place fashion, and will only

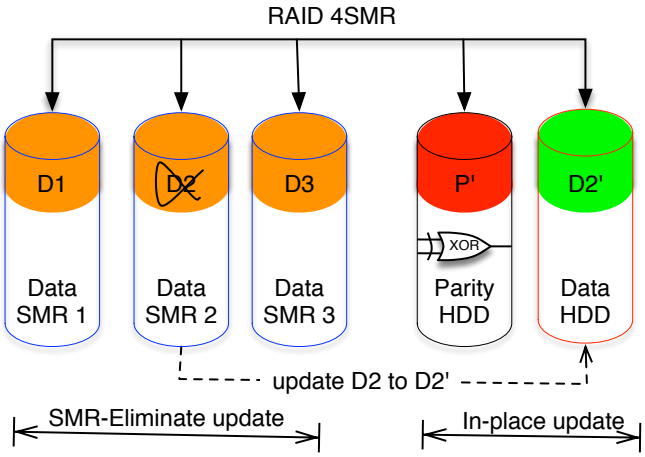


Figure 6.2: When one of the blocks in shingled array is updated, the data on SMR disks will be left unchanged, but the updated block will be written into Data HDD and the corresponding parity block is recalculated.

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 Figure 6.2). 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 [6].

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 Figure 6.3. This will have a slightly

negative impact on overall system reliability, as it creates an interdependency 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).

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 load as well.

Assuming a read is randomly distributed over the range of all blocks and we anticipate 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 the read performance of a standalone RAID 4SMR. Typical read speed for a currently available SMR disk is around 150 MB/s (Seagate 8TB SMR drive [58]). Since RAID 4SMR data is distributed over 3 Data SMR disks, the expected read speed of a RAID 4SMR subsystem is 3 x 150 MB/s or 450 MB/s. With a latest SSD disk (the read speed is around 500 MB/s - Samsung SSD 850 EVO [54]) used in place of the Data HDD disk (which we recommended), the Data HDD should not be a performance bottleneck.

In Figure 6.4 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 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 write to a unique block or update data of a block. If it is the first time we write to the block, we can simply write data to the next available block in one of the Data SMRs[1-3] (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[1-3] disk, 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 3 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 3 data SMRs[1-3] 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.

6.1 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 mapping table will be able to collect all required blocks to get 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 data

Figure 6.3: RAID 4SMR can be chained together to form a bigger array.

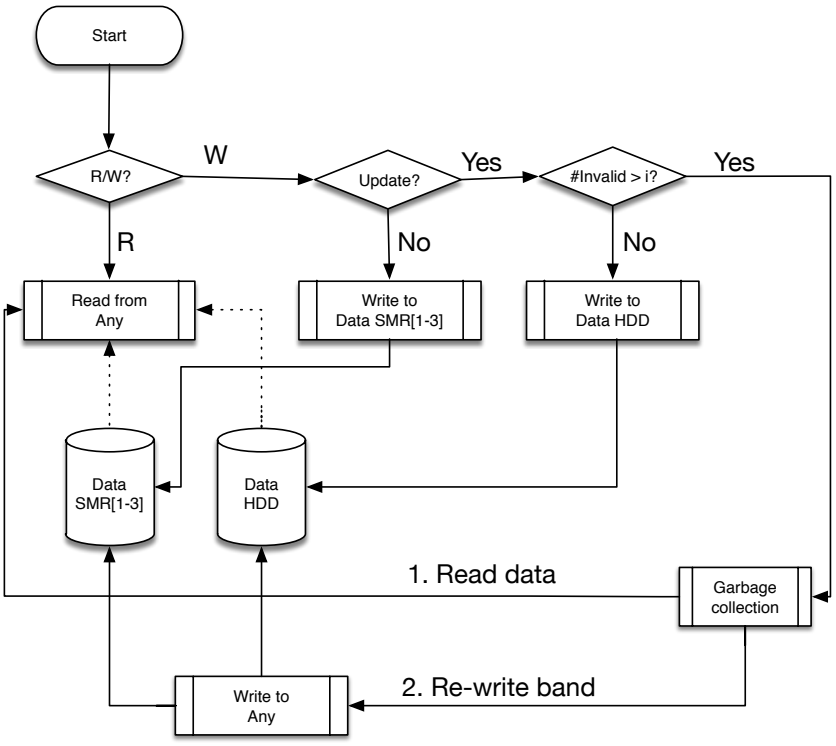


Figure 6.4: Flowchart of how RAID 4SMR works.

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 are 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. Should the crucial Data HDD fail, data in it can be restored from the other three Data SMRs and the parity HDD.

6.2 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 zero and one. The higher value is the better.

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 3 SMR disks (without update) in an array of 5 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 4 disks (all 3 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:

$$1 - \frac{1+c}{n} \leq \text{space efficiency} \leq 1 - \frac{c}{n} \tag{6.1}$$

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 (Figure 6.3), we anticipate the efficiency is in line with RAID 4. Let us say we are chaining 20 RAID 4SMR systems, n will be 81 (4 disks for each RAID 4SMR system and only 1 Data HDD is needed), c will be 20, and the efficiency has a range from 74% to 75%.

6.3 RAID 4SMR Reliability

In this section 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 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

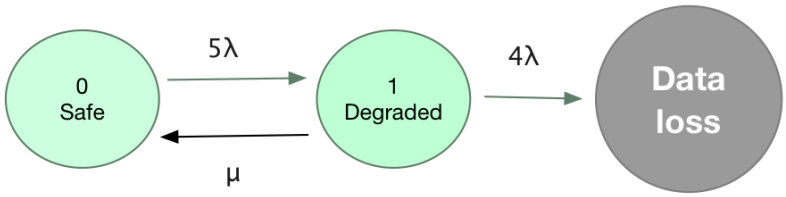


Figure 6.5: State transition probability diagram - Markov Chain for RAID 4SMR.

is exponentially distributed with rate μ.

To simplify the calculation, we analyze a simple RAID 4SMR array with 5 disks. The Markov Chain diagram in Figure 6.5 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 5 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, a failure of a second disk would bring the array to data loss state.

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

$$\frac{dp_0(t)}{dt} = -5\lambda p_0(t) + \mu p_1(t) \tag{6.2}$$

$$\frac{dp_1(t)}{dt} = 5\lambda p_0(t) - (4\lambda + \mu) p_1(t) \tag{6.3}$$

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:

$$sp_0^*(s) = -5\lambda p_0^*(s) + \mu p_1^*(s) + 1 \tag{6.4}$$

$$sp_1^*(s) = 5\lambda p_0^*(s) - (4\lambda + \mu)p_1^*(s) \tag{6.5}$$

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

$$MTTDL = \sum_i p_i^*(0) \tag{6.6}$$

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

$$MTTDL = \frac{\mu + 9\lambda}{20\lambda^2} \tag{6.7}$$

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

$$MTTF = \frac{1}{\lambda} \tag{6.8}$$

$$MTTR = \frac{1}{\mu} \tag{6.9}$$

6.4 RAID 4SMR with Garbage Collection Evaluation

In previous sections, we showed how RAID 4SMR [28] works without garbage collection, which is an ideal situation as we do not have to reclaim the freed blocks. In this section, 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:

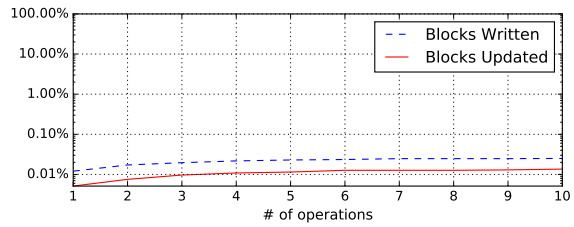
if *totalInvalidBlocksInBands*() >*pre-defined-i* **then**
    $buf \leftarrow getData(DataSMR[1-3], DataHDD)$;
    deleteOldBlocksInDataHDD(); // blocks stored in buf;
    rewriteSMRBands(buf);
end

In order to perform garbage collection, we added the support for reclaiming freed blocks in our simulation. This garbage collection process will be triggered when we have enough i number of freed blocks, which is a pre-defined value. Value of i is in the range of $0 < i < 3 * band\_size$. The highest value means all blocks in three bands across 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.

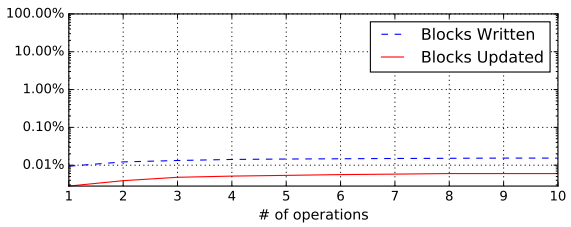
Using the collection of workloads we gathered, we mixed those workloads in the same category together to form newly-merged workloads to simulate the multi-user environments. Although these new workloads have the same name, they may not have the same characteristics as standalone workloads.

We have plotted the new workload's percentage of written and updated blocks in Figure 6.6. 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 showed a variation of the percentage of updates across and within workload types.

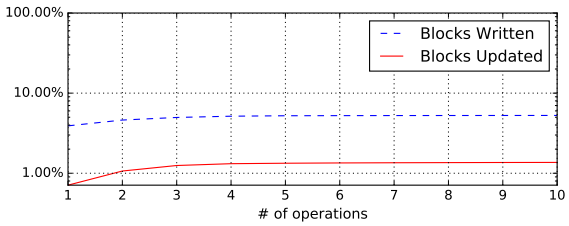
As an SMR drive cannot simply perform an update in-place, we paid particular attention to block update operations. It is these operations which, unless they happen to fall at the last track of a band, would require us to address the update-in-place restriction. Our analysis differs from prior art [2] 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



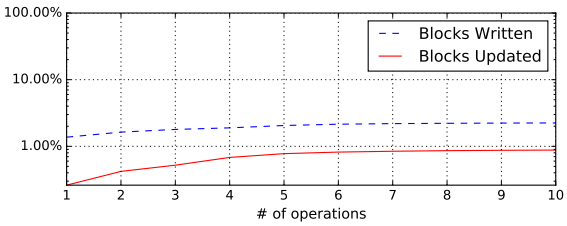
(a) NASA: 0.031% Writes, 0.019% Updates.



(b) RSRCH: 0.016% Writes, 0.006% Updates.



(c) VMWare: 5.286% Writes, 1.382% Updates.



(d) WEB: 2.387% Writes, 1.016% Updates.

Figure 6.6: Four merged workloads from the same category.

writes or updates. At each data point, we plot the percentage of all blocks that were written 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, 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 section, 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 read/write performance max 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 as much as 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 count for the first 3 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 4 disks. The result table's (Table 6.1) last column shows that RAID 4SMR always perform 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 6.1) in the first column, we are confident that RAID 4SMR gc is usable in the real world with different type of workloads. The max 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

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%

Table 6.1: Ratio of number of block movements for RAID 4, RAID 4SMR, and RAID 4SMR gc with SMR disk

movements compared to standard RAID 4. While deploying SMR disks in a chain of RAID 4SMR, the benefits will be even more profounced 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 number of block movements.

Chapter 7

Future Work

To adopt SMR disks with RAID arrays, we start 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 study to evaluate the impact of SMR in more specific contexts. A kernel driver for this RAID 4SMR with garbage collection is needed to adapt the design to the real world. Also, a possibility of a fabric interconnects to the individual drives using erasure coding techniques is considered for a new development since disk performance is quite unpredictable in large-scale data centers. Furthermore, one can increase the reliability of RAID 4SMR by using extra parity disks in a two-dimensional RAID array [44].

Chapter 8

Conclusion

We started out evaluating an SMR disk with our simulation with many different traces. We found that without a proper scheme, SMR disks are almost unusable in an array configuration. With SMR disks, it is easy to assume that any workload that is heavily biased towards writes would be problematic. One might think that it leads to increased activity to compensate for the inability to update most tracks in place. Our workload analysis has shown otherwise. In fact, it would appear that considering the percentage of blocks that experience updates is at least as important as 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. While 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, we found this metric to be largely correlated when track or band movements were considered instead. We therefore believe it to be a uniform metric across workloads and the eventual form of the devices. Our workload analysis leads us to the following conclusions:

1. Significant update workloads usually result in a greater penalty for attempts to emulate update in-place.

2. Logging and remapping approaches, when implemented with minimal overhead,

can result in considerable performance improvements for SMR disks that employ them when compared to traditional disks that do not.

3. Logging and remapping approaches can also result in tremendous performance penalties when faced with a largely referential, read-dominated workload.

4. Enhancing a shingled-write disk with additional NVRAM storage to improve performance can require very little memory to achieve dramatic improvements to the performance of some of the most problematic workloads, but again, this appears to be a workload-specific phenomenon.

Ultimately, it may be that the effective management and identification of I/O workloads becomes one of the most important challenges to effective deployment and adoption of shingled-write disks and the continued growth of mass storage densities.

Later, we have offered our first experimental results evaluating the behavior of SMR disks when used as part of an array, and the impact of increasingly interleaved workloads from different sources. While our previous results in [31] 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. Directing different workloads to different devices can be aided by existing efforts on workload differentiation and tagging [40], and would be a simple way to avoid heavily interleaved workloads.

We proposed RAID 4SMR with SMR disks in places of data disks in a RAID 4 array. Contrary to popular belief that a SMR drive is only suitable for archival storage and would perform worse in RAID arrays, our evaluation shows 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. In our experiments, we compared our proposed RAID 4SMR scheme, not just to a traditional update-in-place arrangement of devices, but also against an idealized form of log-structuring which avoids the need to update in place. We have demonstrated improvement even upon this scheme, which is optimal in terms of log arrangements, but is unlike our scheme in that it is not treating the individual devices as SMR disks [29]. By using the same simulation, we show that RAID 4SMR with garbage collection scheme clearly demonstrates 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%.

.

Bibliography

[1] A. Aghayev, M. Shafaei, and P. Desnoyers. Skylight—A Window on Shingled Disk Operation. *ACM Transactions on Storage (TOS)*, 11(4):16–28, Nov. 2015.

[2] A. Amer, D. D. E. Long, E. L. Miller, J.-F. Paris, and T. Schwarz. Design issues for a shingled write disk system. In *26th IEEE Symposium on Mass Storage Systems and Technology*, pages 1–12, 2010.

[3] A. Amer, J. F. Paris, T. Schwarz, V. Ciotola, and J. Larkby-Lahet. Outshining mirrors: Mttdl of fixed-order spiral layouts. In *Fourth International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI 2007)*, pages 11–16, Sept 2007.

[4] BackBlaze. `http://www.backblaze.com`, Jan. 2019.

[5] Y. Casutto, M. Sanvido, C. Guyot, D. Hall, and Z. Bandic. Indirection systems for shingled-recording disk drives. In *26th IEEE Symposium on Mass Storage Systems and Technology*, pages 1–14, May 2010.

[6] S. Chaarawi, J.-F. Paris, A. Amer, T. Schwarz, and D. Long. Using a shared storage class memory device to improve the reliability of raid arrays. In *Petascale Data Storage Workshop (PDSW), 2010 5th*, pages 1–5. IEEE, 2010.

[7] S. H. Charap, P.-L. Lu, and Y. He. Thermal stability of recorded information at high densities. *IEEE Transactions on Magnetics*, 33(1):978–983, Jan. 1997.

[8] H. Dai, M. Neufeld, and R. Han. Elf: An efficient log-structured flash file system for micro sensor nodes. In *ACM Conference on Embedded Networked Sensor Systems*, pages 176–187, 2004.

[9] DropBox. `http://www.dropbox.com`, Jan. 2019.

[10] R. Finlayson and D. Cheriton. Log files: an extended file service exploiting write-once storage. In *SOSP '87: Proceedings of the eleventh ACM Symposium on Operating systems principles*, pages 139–148, 1987.

[11] G. Gibson and M. Polte. Directions for shingled-write and two-dimensional magnetic recording system architectures: Synergies with solid-state disks. Technical report, Carnegie Mellon University Parallel Data Lab, May 2009. CMU-PDL-09-014.

[12] Google Cloud Storage. `https://cloud.google.com/storage`, Jan. 2019.

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

[14] K. M. Greenan, J. S. Plank, and J. J. Wylie. Mean Time to Meaningless: MTTDL, Markov Models, and Storage System Reliability. *HotStorage*, 2010.

[15] D. Hall, J. H. Marcos, and J. D. Coker. Data Handling Algorithms For Autonomous Shingled Magnetic Recording HDDs. *IEEE Transactions on Magnetics*, 48(5):1777–1781, 2012.

[16] W. He and D. Du. Novel Address Mappings for Shingled Write Disks. *HotStorage*, 2014.

[17] W. He and D. H. Du. SMaRT: An approach to shingled magnetic recording translation. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pages 121–134, Santa Clara, CA, 2017. USENIX Association.

[18] D. Hitz, J. Lau, and M. Malcolm. File system design for an NFS file server appliance. In *Winter 1994 USENIX Conference*, pages 19–19, 1994.

[19] C. Jin, W. Xi, Z. Ching, F. Huo, and C. Lim. Hismrfs: A high performance file system for shingled storage array. In *IEEE 30th Symposium on Mass Storage Systems and Technologies, (MSST)*, pages 1–6, 2014.

[20] S. N. Jones, A. Amer, E. L. Miller, D. D. E. Long, R. Pitchumani, and C. R. Strong. Classifying data to reduce long term data movement in shingled write disks. In *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–9. IEEE.

[21] Kadekodi, Saurabh, Pimpale, Swapnil, and Gibson, Garth A. Caveat-Scriptor: Write Anywhere Shingled Disks. 2015.

[22] P. Kasiraj, R. New, J. de Souza, and M. Williams. System and method for writing data to dedicated bands of a hard disk drive. United States Patent 7490212, 2009.

[23] J. Kohl, C. Staelin, and M. Stonebraker. Highlight: Using a log-structured file system for tertiary storage management. In *Usenix Conference*, pages 435–448, 1993.

[24] A. R. Krishnan, R. Radhakrishnan, and B. Vasic. LDPC decoding strategies for two-dimensional magnetic recording. In *IEEE Global Communications Conference*, pages 1–5, Nov 2009.

[25] A. R. Krishnan, R. Radhakrishnan, B. Vasic, A. Kavcik, W. Ryan, and F. Erden. Two-dimensional magnetic recording: Read channel modeling and detection. In *IEEE International Magnetics Conference*, volume 45, pages 3830–3836, Oct 2009.

[26] M. Kryder, E. Gage, T. McDaniel, W. Challener, R. Rottmayer, G. Ju, Y.-T. Hsia, and M. Erden. Heat assisted magnetic recording. *Proceedings of the IEEE*, 96(11):1810–1835, Nov. 2008.

[27] M. H. Kryder and C. S. Kim. After hard drives – what comes next? *IEEE Transactions on Magnetics*, 45(10):3406–3413, Oct. 2009.

[28] Q. M. Le, A. Amer, and J. Holliday. SMR Disks for Mass Storage Systems. In *MASCOTS 2015, IEEE 23rd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Atlanta, GA, USA, October 5-7, 2015*. IEEE, 2015.

[29] Q. M. Le, A. Amer, and J. Holliday. Smr disks for mass storage systems. In *Proceedings of the 2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '15, pages 228–231, Washington, DC, USA, 2015. IEEE Computer Society.

[30] Q. M. Le, J. Holliday, and A. Amer. *FAST'12: The Peril and Promise of Shingled Disk Arrays: how to avoid two disks being worse than one - WiP.*

[31] Q. M. Le, K. SathyanarayanaRaju, A. Amer, and J. Holliday. Workload impact on shingled write disks: all-writes can be alright. In *MASCOTS*, pages 444–446, 2011.

[32] D. Le Moal, Z. Bandic, and C. Guyot. Shingled file system host-side management of Shingled Magnetic Recording disks. In *2012 IEEE International Conference on Consumer Electronics (ICCE)*, pages 425–426. IEEE, 2012.

[33] S.-W. Lee and B. Moon. Design of flash-based dbms: an in-page logging approach. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 55–66, 2007.

[34] C. I. Lin, D. Park, W. He, and D. H. C. Du. H-SWD: Incorporating Hot Data Identification into Shingled Write Disks. *2012 IEEE 20th International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 321–330, 2012.

[35] W. Liu, D. Feng, L. Zeng, and J. Chen. Understanding the SWD-based RAID System. In *2014 International Conference on Cloud Computing and Big Data (CCBD)*, pages 175–181. IEEE, 2014.

[36] D. Lomet. The case for log structuring in database systems. In *International Workshop on High Performance Transaction Systems*, 1995.

[37] Z.-W. Lu and G. Zhou. Design and Implementation of Hybrid Shingled Recording RAID System. In *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing*, pages 937–942. IEEE, 2016.

[38] K. Matsumoto, A. Inomata, and S. Hasegawa. Thermally assisted magnetic recording. *Fujitsu Scientific and Technical Journal*, 42(1):158–167, Jan. 2006.

[39] J. Matthews, D. Roselli, A. Costello, R. Wang, and T. Anderson. Improving the performance of log-structured file systems with adaptive methods. In *ACM Symposium on Operating Systems Principles*, pages 238–251, 1997.

[40] M. Mesnier, F. Chen, T. Luo, and J. B. Akers. Differentiated storage services. In *Proceedings of ACM SOSP*, 2011.

[41] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. In *FAST'08: Proceedings of the 7th conference on USENIX Conference on File and Storage Technologies*, pages 10:1–10:23, 2008.

[42] NASA. `http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html`, Mar. 2011.

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

[44] J. F. Paris, V. Estrada-Galinanes, A. Amer, and C. Rincon. Using entanglements to increase the reliability of two-dimensional square raid arrays. In *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Dec 2017.

[45] J.-F. Paris, T. Schwarz, A. Amer, and D. D. E. Long. Highly reliable two-dimensional raid arrays for archival storage. In *31st IEEE International Performance Computing and Communications Conference*, Dec. 2012.

[46] J. F. Paris, T. Schwarz, and D. Long. When MTTDLs are not good enough: Providing better estimates of disk array reliability. *Proceedings of the 7th International Information and Telecommunication Technologies Symposium*, 2008.

[47] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, SIGMOD '88, pages 109–116, New York, NY, USA, 1988. ACM.

[48] D. Pease, A. Amir, L. V. Real, B. Biskeborn, and M. Richmond. Linear tape file system. In *26th IEEE Symposium on Mass Storage Systems and Technology*, pages 1–8, May 2010.

[49] Z. Peterson and R. Burns. Ext3cow: A time-shifting file system for regulatory compliance. *Trans. Storage*, 1(2):190–212, May 2005.

[50] M. Rosenblum. *The Design and Implementation of a Log-structured File System*. PhD thesis, UC Berkeley, 1992.

[51] M. Rosenblum and J. Ousterhout. The design and implementation of a log-structured file system. *Operating Systems Review*, 25(5):1–15, Oct. 1991.

[52] J. B. Rosie Wacha, Scott A. Brandt and C. Maltzahn. *FAST'10: RAID4S: Adding SSDs to RAID Arrays - WiP*.

[53] R. E. Rottmeyer, S. Batra, D. Buechel, W. A. Challener, J. Hohlfeld, Y. Kubota, L. Li, B. Lu, C. Mihalcea, K. Mountfiled, K. Pelhos, P. Chubing, T. Rausch, M. A. Seigler, D. Weller, and Y. Xiaomin. Heat-assisted magnetic recording. *IEEE Transactions on Magnetics*, 42(10):2417–2421, Oct. 2006.

[54] Samsung Electronics Co., Ltd. SSD Samsung 850 EVO. `https://www.samsung.com/semiconductor/minisite/ssd/product/consumer/850evo`, 2019. [Online; accessed 02-Jan-2019].

[55] C. K. Sann, R. Radhakrishnan, K. Eason, R. M. Elidrissi, J. Miles, B. Vasic, and A. R. Krishnan. Channel models and detectors for two-dimensional magnetic recording (TDMR). volume 46, pages 804–811. IEEE, 2010.

[56] T. Schwarz, A. Amer, T. Kroeger, E. L. Miller, D. D. E. Long, and J.-F. Paris. Resar: Reliable storage at exabyte scale.

[57] Seagate Technology LLC. Introducing Seagate SMR. https://www.seagate.com/tech-insights/breaking-areal-density-barriers-with-seagate-smr-master-ti, Jan. 2019. [Online; accessed Jan-2019].

[58] Seagate Technology LLC. SMR Drive 8TB - ST8000AS0002. https://www.seagate.com/www-content/product-content/hdd-fam/seagate-archive-hdd/en-us/docs/archive-hdd-ds1834-5c-1508us.pdf, 2019. [Online; accessed 02-Jan-2019].

[59] M. Selzer, K. Bostic, M. McKusick, and C. Staelin. An implementation of a log-structured file system for UNIX. In *Winter Usenix Conference*, pages 3–3, 1993.

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

[61] I. Tagawa and M. Williams. High density data-storage using shingle-write. In *Proceedings of the IEEE International Magnetics Conference*, May. 2009.

[62] Western Digital. https://support.wdc.com/knowledgebase/answer.aspx?ID=26014, Jan. 2019.

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

[64] M. Yang, Y. Chang, F. Wu, T. Kuo, and D. H. C. Du. Virtual persistent cache: Remedy the long latency behavior of host-aware shingled magnetic recording drives. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 17–24, Nov 2017.

[65] X. Zhang, D. Du, J. Hughes, and R. Kavuri. Hptfs: A high performance tape file system. In *26th IEEE Symposium on Massive Storage Systems and Technology*, May. 2006.

[66] J.-G. Zhu, X. Zhu, and Y. Tang. Microwave assisted magnetic recording. *IEEE Transactions on Magnetics*, 44(1):125–131, Jan. 2008.

Glossary of Terms[1]

HDD hard disk, hard drive, or fixed disk, is an electromechanical data storage device that uses magnetic storage to store and retrieve digital information using one or more rigid rapidly rotating disks (platters) coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored or retrieved in any order and not only sequentially. 35

log-structured file system is a file system in which data and metadata are written sequentially to a circular buffer, called a log. The design was first proposed in 1988 by John K. Ousterhout and Fred Douglis and first implemented in 1992 by John K. Ousterhout and Mendel Rosenblum. 4

parity a parity bit, or check bit, is a bit added to a string of binary code to ensure that the total number of 1-bits in the string is even or odd. Parity bits are used as the simplest form of error detecting code. 37

RAID (originally redundant array of inexpensive disks, now commonly redundant array of independent disks) is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both. 35

RAID 4 consists of block-level striping with a dedicated parity disk. As a result of its layout, RAID 4 provides good performance of random reads, while the performance of random writes is low due to the need to write all parity data to a single disk. 35

[1]From wikipedia.org

RAID 5 consists of block-level striping with distributed parity. Unlike in RAID 4, parity information is distributed among the drives. It requires that all drives but one be present to operate. Upon failure of a single drive, subsequent reads can be calculated from the distributed parity such that no data is lost. RAID 5 requires at least three disks. 35

RAID 6 extends RAID 5 by adding another parity block; thus, it uses block-level striping with two parity blocks distributed across all member disks. 35

SSD is a solid-state storage device that uses integrated circuit assemblies as memory to store data persistently. It is also sometimes called a solid-state disk, although SSDs do not have physical disks. 4

Acronyms

CMR Conventional Magnetic Recording. 3

COW Copy-On-Write. 32

HAMR Heat-Assisted Magnetic Recording. 7

HDD Hard Disk Drive. 3

I/O Input/Output. 1

LBA Logical Block Addressing. 16

MAMR Microwave-Assisted Magnetic Recording. 7

MTTDL Mean Time To Data Loss. 43

MTTF Mean Time To Failure. 45

MTTR Mean Time To Repair. 45

NVRAM Non-Volatile Random-Access Memory. 15

PBA Physical Block Address. 16

RAID Redundant Array of Independent Disks. 35

SMR Shingled Magnetic Recording. 1

SSD Solid-State Drive. 4

TDMR Two-Dimensional Magnetic Recording. 1

WAFL Write-Anywhere File Layout. 32

WORM Write One Read Many. 36