Performance Evaluation of Host Aware Shingled Magnetic Recording (HA-SMR) Drives

Fenggang Wu, Ziqi Fan, Ming-Chang Yang, Baoquan Zhang, Xiongzi Ge and David H.C. Du

Abstract—Shingled Magnetic Recording (SMR) drives can benefit large-scale storage systems by reducing the Total Cost of Ownership (TCO) and meeting the challenge of the explosive data growth. Among all existing SMR models, Host Aware SMR (HA-SMR) looks the most promising for its backward compatibility with legacy I/O stacks and capability of using the new SMR-specific APIs to support the host I/O stack optimization. Building HA-SMR drive based storage systems calls for a deep understanding of the drives performance characteristics. To accomplish this, we conduct in-depth performance evaluations on HA-SMR drives with a special emphasis on the performance implications of the SMR-specific APIs and how these drives can be deployed in large storage systems. We explore various workload types and parameters and discover both favorable and adverse use-cases for HA-SMR drives. We also investigate the drives performance under legacy production environments using real-world enterprise traces. To remedy the potential severe performance degradation in certain conditions, we propose to add a novel host-controlled buffer that can help to reduce the severity of the HA-SMR performance under unfriendly I/O access patterns. Without a detailed comprehensive design, we show the potential of the host-controlled buffer by a use case study in this paper.

Index Terms—Host Aware SMR; Characterization; System Implication; H-Buffer.

1 Introduction

Big data era calls for Petabyte storage systems with affordable Total Cost of Ownership (TCO). To meet this challenge, hard disk drive industry pushes very hard in developing drives with higher areal density to cut down the dollars per gigabyte as well as the power consumption. As perpendicular magnetic recording (PMR) [1] technology used by the conventional HDDs is reaching its areal density limit, other new recording technologies such as Heat-Assisted Magnetic Recording (HAMR) [2] [3], Bit-Patterned Media (BPM) [4] and SMR (Shingled Magnetic Recording) [5] [6] [7] are investigated to overcome this barrier. While both HAMR and BPM are not commercialized yet due to certain technical difficulties, SMR drives have already been available recently to general customers [8].

Since the width of the write head is wider than that of the read head, in SMR technology data are written partially overlapped with the adjacent tracks in a shingled fashion while the data can still be read from the uncovered portion of data tracks. Consequently, updating existing data blocks in SMR drives needs more caution as it may destroy data on overlapping data tracks. As a solution, writing data sequentially and update by Read-Modify-Write in the shingled written space will protect the data from being destroyed. Instead of having a huge shingled written space, it is better to have multiple individual ones, a.k.a. *bands*, separated by guard tracks to increase flexibility. Accroding to the T10 [9] and T13 [10] standards, the physical SMR bands are abstracted as logical *zones* representing consecutive non-overlapping LBA ranges. Each of the zone is designed to be written in a log structure manner, and is associated with a *write pointer* indicating the location in the zone on which the next write should target.

There are three types of SMR drives existing, i.e. *drive managed*, *host aware*, and *host managed* SMR drives [11]. The host (a server) interacts with the SMR drives in different ways according to their types.

Manuscript received MMM DD, YYYY; revised MMM DD, YYYY.

Drive Managed SMR (DM-SMR) drives buffer the updating I/O traffic in a *media cache* (or *persistent cache*, one portion of the disk media) and later migrate those data blocks back to the intended locations (LBAs in the targeted zones) by a *media cache cleaning* process. This persistent cache as well as the cleaning operations are hidden from the host, therefore, DM-SMR drives interact with the host in the same way as a conventional non-SMR drives [11] [12]. Host Aware SMR (HA-SMR) drives also use the media cache buffering and cleaning to handle the updating I/O traffic, or *non-sequential write* requests in the zoned block device terminology (i.e. write requests not beginning at the write pointer of a zone) [9] [10]. Further, HA-SMR drives expose more internal data layout information, such as the number of *zones* and the start LBA/the write pointer/various flags of each zone, to the host via *zoned block APIs* [13] [9] [10]. Host Managed SMR (HM-SMR) drives expose the same zoned block APIs as HA-SMR drives do, but they do not accept non-sequential writes and hence do not need the persistent cache component [13] [9] [10].

In this paper, we choose HA-SMR drives as our research subject because we believe HA-SMR is expected to be more popular than the other two models due to the following reasons:

- Compared to HM-SMR model, HA-SMR is more backward compatible with legacy software, so SMR drive users can still benefit from the lower TCO without revamping their applications or I/O stack; and
- Compared to DM-SMR model, HA-SMR provides the opportunity of zone-aware software optimization

for a higher and more predictable performance by using zoned block APIs which can provide detailed zone information to the host.

There are fewer existing studies that addresses SMR performance characterization. The most closest one – Skylight [12] – focuses on uncovering the internal structural information of the "black box" **DM-SMR** model. A thorough investigation of **HA-SMR** drives is still missing, especially the performance implications of the HA-SMR model due to its unique zoned block APIs.

We carry out an in-depth performance evaluation on several HA-SMR sample drives with a special emphasize on the performance characterization regarding the newly-introduced zoned block APIs. For examples, we study the performance impact of the zone sequential writes and non-sequential writes regarding to the number of zones that are in the writing working set (open zone and non-sequential zone issues). Besides, as media cache cleaning efficiency is the key factor to determine the non-sequential writes performance, we investigate how different workloads could affect the media cache cleaning efficiency. Based on the testing results, we summarize the system implications of the unique HA-SMR performance characteristics for building large scale storage systems. Further, we replay real world traces against the HA-SMR drives and explain the results using the knowledge we have obtained from the performance evaluation.

Based on our observations of SMR drive performance and by fully exploiting the HA-SMR model, we further propose a *host-controlled indirection buffer* (called *H-Buffer*) which has the potential to improve the performance of HA-SMR drives by combining the strength of both the drive and the host and transferring SMR-unfriendly I/O traffic patterns into more favorable ones. Without a comprehensive design, we have demonstrated the usage and benefit of the H-Buffer in a simple use case study.

The rest for the paper is organized as follows. In Sec. 2 we introduce the preliminaries of the HA-SMR drives. Sec. 3 describes the measurement system and experimental setup. Detailed tests and their corresponding results are presented in Sec. 4. We discuss the system implications of using HA-SMR drives in Sec. 5. Real world trace replay results are analyzed in Sec. 6. The proposed H-Buffer is described in Sec. 7. Sec. 8 presents a use case study of the H-Buffer. We discuss the related work in Sec. 9 and conclude the paper in Sec. 10.

2 Preliminaries

2.1 SMR Technology

In order to break the 1Tb per square inch areal density limit [14] of the traditional Perpendicular Magnetic Recording (PMR) [1], in Shingled Magnetic Recording (SMR), neighboring data tracks are written overlapped with each other in a shingled fashion [5] [6] [7]. As opposed to SMR, the traditional recording technique is therefore referred to as Conventional Magnetic Recording (CMR) [12].

As updating data directly on a shingled data track may destroy other data blocks on overlapped tracks, the data blocks on these overlapped tracks need to be read out before updating and rewritten afterwards. However, rewriting data blocks to the overlapped tracks will potentially destroy the data blocks in their overlapped tracks further. This in turn calls for rewriting data on further overlapped tracks and leads to a cascading write amplification. To limit such write amplification, guard tracks are inserted after every few tracks to stop the data track update propagation. These guard tracks divide the whole disk space into many *bands*. Each band can be individually modified without interfering with each other. Clearly, bands with a small number of tracks (small bands) will cause less write amplification, but it will have less gain in areal density as well. The band size of the current HA-SMR drives is a compromise of these two factors.

Some of the SMR drives also reserve a small portion of the disk space as *media cache* (also called *persistent cache*) which buffers the non-sequential write requests from the host. From time to time the drive will have to migrate the buffered data blocks from the media cache to their intended shingled bands (*media cache cleaning*). In order to do this, a Shingled Translation Layer (STL) is needed to map the Logical Block Addresses (LBAs) to the current Physical Block Addresses (PBAs) [15] [16] [17] [18].

2.2 Zone Modeling of SMR Drives

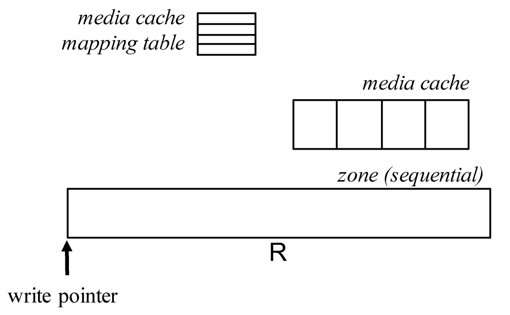
T10 ZBC [9] and T13 ZAC [10] standards are currently under revision aiming to define the interacting models as well as the SCSI/ATA command extensions for the SMR drives. The standards abstract an SMR drive as a collection of *zones* which are logical non-overlapping collections of consecutive LBAs. The notion of a zone in the standards largely corresponds to the band concept in the SMR drive implementation. In the rest of this paper, we will use band and zone interchangeably.

There are two types of zones: *conventional zones* and *write pointer zones* which are conceptually mapped to bands consisting of CMR tracks and SMR tracks respectively. A write pointer zone is designed to be written in a log structure. It has an associated *write pointer* which indicates an LBA within the zone where the next write should target. By contrast, conventional zones have no write pointers. Majority of the zones in SMR drives are shingled. So, unless other stated, we use "zones" to refer to "write pointer zones" in the rest of the paper.

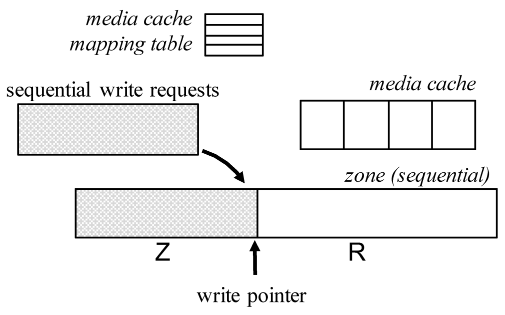
2.3 Sequential and Non-Sequential Writes

Applications can issue a write operation to a zone either at the write pointer or not and the corresponding write operation is defined as either *sequential write* or *non-sequential write* correspondingly. Note that this "sequential" is different from the traditional term of sequential I/O (as opposed to random I/O) that depends on the consecutiveness of the written LBA, the inter-arrival time, the size of data requested, and etc. [19]. In this paper we use the term sequential and non-sequential writes to refer to the write requests happening at and not at the write pointer respectively.

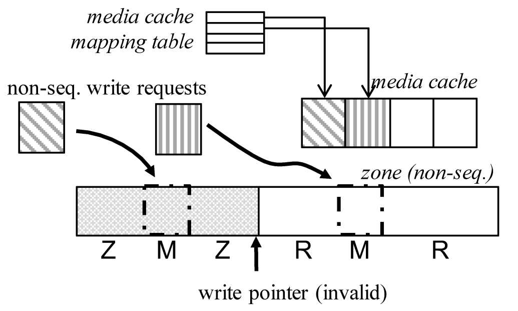
There are two types of write pointer zones, namely sequential write *preferred* zones and sequential write *required* zones. In both of them, sequential writes will go to the targeted LBAs directly and the write pointer will be advanced by the size of blocks successfully written. The write pointer is initially placed at the beginning of the zone (zone is empty). If there are continuous sequential writes, the write



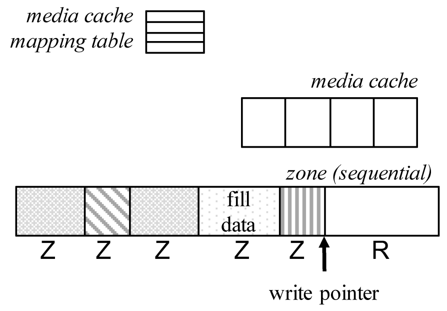
(a) Zone initially empty



(b) After several sequential write requests



(c) After two non-sequential write requests



(d) After media cache cleaning this zone

Fig. 1. Zone operations illustrated. (Z, M, R means reading from those LBAs will return data from Zone, Media cache or disk RAM.)

TABLE 1
Disk Models for SMR Drive

	DM	HA	HM
Zoned Block Dev.	No	Yes	Yes
Media Cache	Needed	Needed	Not Needed
Conventional Zone	N/A	Optional	Optional
Seq-write-pref Zone	N/A	Mandatory	Not supported
Seq-write-req Zone	N/A	Not supported	Mandatory

pointer will advance accordingly until it reaches the end of the zone (zone is full). Sequential write required zone will reject any non-sequential writes. Conversely, sequential write preferred zone accepts non-sequential writes by temporarily buffer the data in the *media cache* and later migrate them back to the intended zones.

2.4 Three SMR Drive Models

There are three models [9] [10] of SMR drives: drive managed (DM), host aware (HA) and host managed (HM), as summarized in Table 1.

A Host Aware SMR (HA-SMR) drive consists of sequential write preferred zones and will accept both sequential and non-sequential writes with the help of the media cache. A Host Managed SMR (HM-SMR) drive is made up of sequential write required zones and will reject any non-sequential write requests. Therefore, such drive cannot be used directly in existing systems without modifying the application software. Both the HA-SMR and HM-SMR drives can optionally have a small number of conventional zones. A Drive Managed SMR (DM-SMR) drive behaves like a conventional non-shingled hard disk drive with no notion of zones. It exposes a linear space of LBAs to the host hiding all the detailed operations of the media cache and the SMR zones from the host. It can be a "drop-in" replacement solution without any modifications to the existing storage systems.

We believe that HA-SMR drive is the most promising among the three. On one hand, similar to HM-SMR drives, HA-SMR drives expose the zoned block APIs to the application developers so that they can design applications that fully respect the shingled constraints. On the other hand, like DM-SMR drives, HA-SMR drives can accept non-sequential writes by buffering them in the media cache and migrate them back later. Therefore, an HA-SMR drive can also be used as a "drop-in" replacement solution. Moreover, HA-SMR has the potential of reduce media cache operation overhead with the help of the zone block APIs. Therefore, in this paper we focus on HA-SMR drives. However, some of the conclusions can be also applied or extended to DM-SMR and HM-SMR drives.

2.5 Data Handling in HA-SMR Drives

In HA-SMR model, a zone is initially empty with the write pointer pointing to the first LBA (Fig. 1a). Before any non-sequential write, a zone is "sequential". That is, all the data that have been written, if there is any, are all issued at the current write pointer (Fig. 1b).

A zone is changed to a "non-sequential" state after a non-sequential write request. Such non-sequential written data is buffered in the media cache and the write pointer is invalidated without moving. The media cache, which is one portion of the disk media, is formatted as a self-describing journal for buffering non-sequentially written data. Subsequent writes to a non-sequential zone are regarded as non-sequential writes and will also be redirected to the media cache (Fig. 1c). Later, a *media cache cleaning* process will migrate all the buffered data back to the targeted zones.

A media cache cleaning algorithm will read out the oldest journaled block (i.e. in a FIFO order) from the media cache and any other blocks in the media cache that belong to the same zone. It will also read out the existing data in the corresponding zone. Then all the data blocks are combined into a consecutive extent and written back to the zone. The zone will then be converted back into a sequential state with its new write pointer properly set. After this, the algorithm can proceed with the next oldest block in the media cache journal [12].

When performing media cache cleaning, if there is any "gap" – LBAs not written yet by the host – before the new write pointer location, synthesized "fill data" will be generated and written to such gap space (Fig. 1d) [20]. This guarantees that the new write pointer will be the LBA right after the highest LBA written by the host.

For a sequential zone (Fig. 1a, 1b, and 1d), reading data from a location before the write pointer location will retrieve data from the drive which may include the disk-resided "fill data" (Fig. 1d), while reading from a location beyond the write pointer will simply return synthesized "fill data" from disk RAM without any accesses to the drive. For a read request to a non-sequential zone, data can be accessed from either the zone, media cache or disk RAM for different cases as shown in Fig. 1c.

2.6 Zoned Block APIs and Zone Open/Close Semantics

The standards for SMR drives augment the existing SCSI/ATA command set with more zone-specific commands [9] [10]. In the host aware model, we can issue read/write commands to the device in the same way as we do to the conventional drives. Besides, there are a few new commands been defined such as OPEN ZONE, CLOSE ZONE, RESET WRITE POINTER, etc. Note that such zone concept and the zoned block APIs do not exist in any storage media before SMR drives.

Each zone has to be opened either *explicitly* or *implicitly* for writing. A zone can be explicitly opened by issuing the OPEN ZONE command or implicitly opened by writing data to the zone. Read operations, however, do not require opening a zone beforehand.

Zone meta-data (write pointer and various flags) that are frequently accessed during read/write operations are kept in the disk RAM. When a write operation issued to the drive updates the zone meta-data in the RAM of the drive, the on-disk copy of the zone meta-data becomes stale and the disk is at risk of losing data. It is critical for the zone meta-data in the drive's RAM to be frequently synchronized back to the platter. To reduce the overhead of such frequent synchronization of zone meta-data, HA-SMR drives have some limited "open zone resource" that can hold the zone meta-data intact even through an unexpected power loss. Such "open zone resource" is allocated for each opened zone and will be reclaimed when the zone is closed [20].

2.7 Open Zone and Non-Sequential Zone Issues

As shown in the standards, there is a recommended maximum number of open zones which corresponding to the limited open zone resources described in Sec. 2.6. From system boot-up, the first few open zone operations only result in resource allocation without causing any disk synchronizations. However, as the host opens more and more zones and the open zone resource is used up, some zones have to be closed to release the open zone resource. Such zone close operations are done either explicitly by the host issuing CLOSE ZONE commands or implicitly by the drive selecting and closing zones on its own. Closing a zone will write back the zone meta-data to the drive incurring expensive disk operations. As a result, if the zone working set for write operations consists of a large number of zones, the write performance is expected to suffer because of the "open zone resource" thrashing (*Open Zone Issue*).

Besides, the standards also indicate a recommended maximum number for non-sequential zones. The reason is that the drive has a limited capability to clean non-sequentially written data from the media cache to the intended zones. When the non-sequential write requests are too many and span over a large number of zones, media cache resources are depleted, cleaning becomes *blocking* (blocking media cache cleaning), and thus hurts the performance of both the writes and reads (*Non-sequential Zone Issue*).

TABLE 2
HA-SMR Drive Specification

Attribute	Value
interface	ATA ZAC
model	host-aware disk model
Seagate model No.	ST8000AS0022-1WL
prototype firmware revision	SN03
logical block: num/size	15628053168 / 512B
physical block: num/size	1953506646 / 4096B
capacity	8001.563 GB
cache/buffer size	16384KB
normal media rotation rate	5980RPM
zone number/size	29809 / 256MB
conventional zones	64 (0-63)
seq-write-pref zones	29745 (64-29808)
seq-write-req zones	Not Applicable
optimal open zone num	128
optimal non-sequential zone num	8

3 Measurement Environment

3.1 SMR Sample Drives and Testing System

We use several Seagate 8TB Host-Aware SMR sample drives for all the tests and performance evaluations. Table 2 summarizes the basic specification of these drives. Note that the drive has a recommended maximum number of 128 open zones and 8 non-sequential zones.

The testing system is a Dell PowerEdge R420 1U Server. It is equipped with two Intel(R) Xeon(R) E5-2407 2.20GHz processors and 32GB DDR3 DIMM memory. Our HA-SMR sample drives are connected to the server via 3 Gbps SATA motherboard connectors. The system is installed with Ubuntu 14.04.3 LTS with Linux kernel version 4.1.6.

3.2 Library and Benchmarking Tools

libzbc [21] is a user-level library developed by HGST to manipulate zoned block devices. The sample drives are compatible with the `r04` branch of *libzbc* `v4.0.0`. For HA-SMR drives, except the zoned block APIs, other commands are backward compatible with the SBC standard [22]. Therefore, normal read and write operations can be supported by HA-SMR drives natively without help of any libraries.

In our tests, we use *libzbc* to get the geometry (zone number and size) of the drive, to monitor the write pointers and the number of non-sequential zones, and to reset the

TABLE 3
Fundamental Parameters Testing Result (Compared with Skylight [12], our result highlight in bold)

Drive Model	**8TB HA (our sample)**	5TB DM [12]	8TB DM [12]
Media cache type	**Disk**	Disk	Disk
Media cache location	**Single, at outer tracks**	Single, at outer tracks	Single, at outer tracks
Media cache size	**25.6GB**	20GB	25GB
Media cache mapping table size	**185,000**	200,000	250,000
Band size	**256 MiB (zone size)**	13-36MiB	15-40MiB
LBA-PBA mapping	**Static**	Static	Static
Cleaning type	**Aggressive, FIFO**	Aggressive, FIFO	Aggressive, FIFO
Cleaning time	**1∼30+ sec/zone**	0.6∼1.6 sec/band	0.6∼1.6 sec/band

zone write pointers when needed. *fio* [23] `v2.6` is used to replay various micro-benchmark traces to the sample drives and to measure performance in terms of latency, throughput, IOPS, run time, etc.

4 Evaluation Results

Our evaluation has a special emphasize on the zoned block APIs and the HA-SMR specific media cache effects on the I/O performance. In this section, we present first the testing results of the fundamental SMR drive parameters, then an investigation on the performance impact of the open zone issue and non-sequential zone issue, and finally a study on how different workloads affect the media cache cleaning efficiency. In these tests, the write-cache and read-ahead of the drives are disabled to exclude the performance interference of the disk RAM and to isolate the investigation of the zoned specific factors.

4.1 Testing for Fundamental SMR Drive Parameters

We carried out tests similar to Skylight [12] to discover the fundamental parameters of the internal structures of the HA-SMR drives including media cache location and size, mapping table size, average per zone cleaning time, as well as the band size, mapping type, etc. The results are summarized in Table 3.

Similarly, our sample drives also do an aggressive and FIFO cleaning. “Aggressive” means that the drive triggers media cache cleaning as soon as the drive becomes idle [12]. Conversely, we have found that the band (zone) size is much larger for our HA-SMR sample drives (256MiB which is actually the zone size) than that of the two DM-SMR drives described in [12] (15-40MiB). Besides, the average per zone cleaning time for our sample drives can be much larger and varies widely. We have an in-depth investigation which explains the reasons of such huge variation in Sec. 4.4.

4.2 Open Zone Issue

Although the drive specifies the recommended maximum number for the open zones, system designers may need to know exactly how the performance is degrading when the recommendation is not followed. In order to provide a reference point for the system designers and to motivate the solutions, we conduct the following measurements to evaluate the open zone issue for both sequential and non-sequential workloads.

We use a light and bursty workload for this test to exclude the media cache cleaning impact and isolate the open zone issue for analysis. Such workload is characterized by long idle time between bursty I/O requests and can be typically found in personal computers and archival systems.

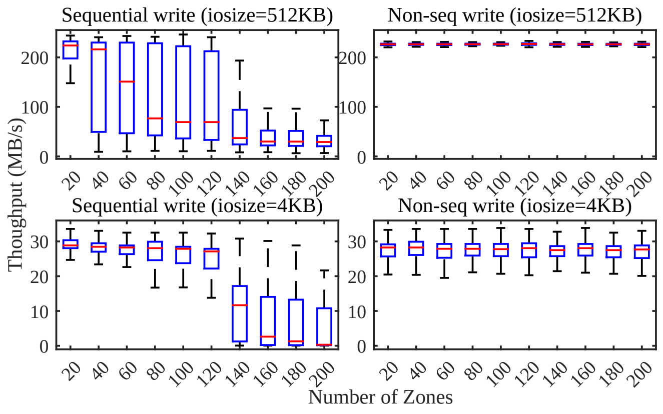


Fig. 2. Throughput v.s. number of open zones.

The test program issues 1000 write requests in a round-robin fashion among a number of zones during each burst. The number of the open zones ranges from 20 to 200. Both sequential writes and non-sequential writes are evaluated. Besides, we test with large ($512KB$) and small ($4KB$) I/O request sizes. The results are summarized as boxplot in Fig. 2. The x-axis of Fig. 2 represents the number of open zones and y-axis represents the throughput. The highest and lowest lines in throughput show the maximum and minimum throughputs respectively of each case based on the number of open zones. The middle rectangular box represents the throughput between the first and the third quartiles. The horizontal line in the middle of each box represents the median throughput. Based on the results, we observe that for sequential writes, the performance drops clearly from 120 zones to 140 zones, which is consistent with the self-reported 128 recommended maximum open zone number. Specifically, the median throughput decreases by 57% when the number of open zones increases from 120 to 140 in the $4KB$ case. We will explore a possible solution for such a performance degradation in Sec. 8.

Surprisingly, the performance of non-sequential writes shows no significant throughput degradation as the open zone number increases. This is probably because non-sequential writes are redirected to media cache and the self-describing journal structure (see Sec. 2.4) guarantees data reliability without using too many “open zone resources”.

4.3 Non-Sequential Zone Issue

The recommended maximum non-sequential zone number indicates a suggested number of zones that can be non-sequentially written in order to limit the media cache cleaning overhead. Nevertheless, legacy zone-unaware workloads could inevitably scatter non-sequential write requests to more zones than the recommended number (8 for our sample drives). Therefore, we still want to study the performance characteristics of the legacy workloads.

In the previous test, we find that a light and bursty *non-sequential* workload experiences no noticeable performance degradation when the open zone number goes beyond the recommend maximum. This is because the idle time is long enough for the idle cleaning to finish before next burst of I/O requests arrives. Therefore, a light and bursty workload cannot show the media cache cleaning impact on I/O performance.

In this test we design a long-run write intensive workload that covers different number of zones. Such workload is able to deplete the media cache resource and forces the drive to do the blocking cleaning while serving write requests. Specifically, a 256KB-sized non-sequential write workload is performed for 2 hours into 128 and 256 zones respectively. The throughput and the number of non-sequential zones are summarized in Fig. 3.

In both 128 and 256 zone cases of Fig. 3, the non-sequential zone number decreases periodically indicating that the drive starts the blocking cleaning while serving the write requests. Based on the throughput and the non-sequential zone number plots, it can be seen that each time the drive starts to clean the media cache, the throughput drops from over $100MB/s$ to a very low rate around $0.1MB/s$ (0.1% of the normal throughput). During the first cleaning cycle, the throughput stays at the low rate for over 25 minutes in the 128-zone case and over 37 minutes in the 256-zone case. This is certainly a very severe degradation of performance that to be concerned by system designers.

We speculate that the low throughput interval will be prolonged when the number of the non-sequential zones increases. As the number of non-sequential zones increases, less space is reclaimed in media cache when the drive cleans one targeted zone (migrating media cache buffered data back to this targeted zone by reading, combining, and rewriting this zone). Therefore, more zones have to be cleaned to reclaim the same media cache space enough for the drive to perform I/O in a normal rate. To verify our speculation, we create an extreme workload in which the non-sequential writes span the whole drive (29808 zones). Consequently, the first blocking cleaning takes so long that the testing program breaks down after about 100 minutes of low-rate interval. A possible explanation is that the cleaning speed is extremely slow and the drive throttles the incoming writes hard for a long duration that causes some timeout faults in the Linux stack. However, such an extreme workload is not expected in a real world environment.

4.4 Media Cleaning Efficiency

From the test in Sec. 4.1, we notice that the average per zone cleaning time varies in a wide range (1~30+s). The average per zone cleaning time is closely related to the media cache cleaning impact on the I/O performance because the faster the drive finishes cleaning, the sooner the drives throughput recovers to the normal level. Here we use the average per zone idle cleaning time as a metrics for the media cache cleaning efficiency, i.e., the higher is the average per zone cleaning time, the lower the cleaning efficiency is.

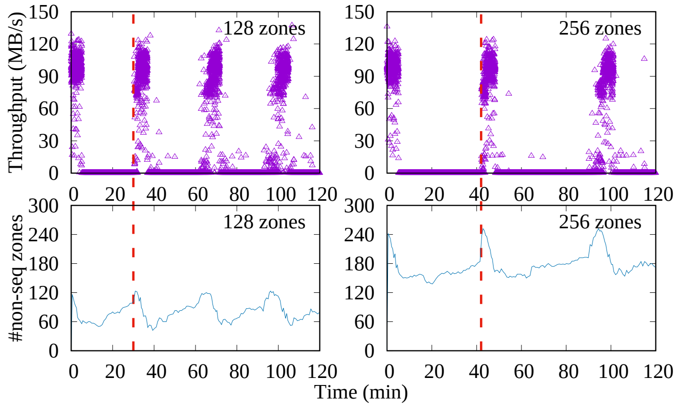


Fig. 3. Throughput and number of non-sequential zones vs time when writing to 128 or 256 zones. Vertical dash lines show the correlation between the throughput and the cleaning process.

We hypothesize that the zone cleaning time depends on the workload characteristics including *the updated ratio, the I/O request location, I/O request size, I/O request number, zone data distribution in media cache*, etc. Here the updated ratio (U) of a zone is defined as the proportion of data that is non-sequentially written within a zone. The I/O request location is defined as the location where the update happens within a zone.

4.4.1 *Update ratio and the I/O request location*

We design the first test to study the impact of the update ratio and the I/O request location on the average per zone cleaning time. The test program updates 1000 zones in a round robin fashion with different update ratios (U=20%, 40%, 60%, 80%, and 100%). Note that such update traffic is a non-sequential write workload. Four different I/O request locations are used: the beginning of the zone (start), the middle of the zone (middle), the end of the zone (end) and random offsets within the zone (rand). The initial state of a zone can be either empty or full. After the replay finishes, we record the idle clean time and divided it by the number of zones (1000) to estimate the average per zone cleaning time (Fig. 4).

From the figure, we can have the following observations: As U increases, the cleaning time goes up accordingly since more data blocks need to be migrated from media cache. If we start from an empty zone, when U increases to 100%, there is actually no difference whichever LBA location we pick within the zone (i.e. start, middle, end or random). From the bar plot we see the four cases converge at $U = 100\%$. The same happens if we start from a full zone.

When writing to an empty zone, the closer the I/O request location is to the zone end, the longer it takes to clean that zone. Note that a random selected LBA also results in the same highest cleaning time. The reason is that when migrating data from media cache back to intended zones, the drive has to write some synthesized fill data to the unwritten areas before the write pointer (Sec. 2.5). Therefore, updating to higher LBA positions in an empty zone results in more extra data to be read/written in the later cleaning process.

By contrast, when writing to a full zone, the closer the I/O request location is to the zone beginning, the longer the cleaning time is. Similarly, a random selected LBA also

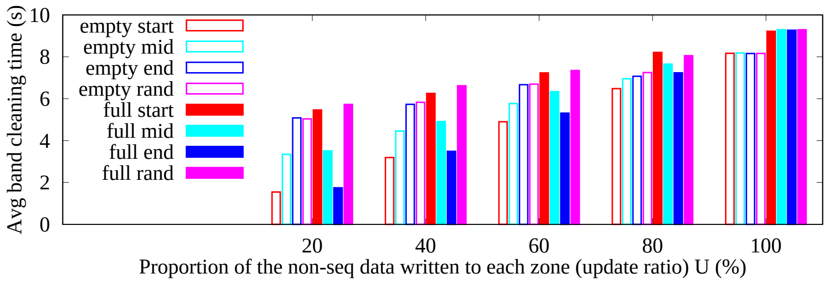


Fig. 4. Impacts of update ratio and the I/O request location on the average per zone cleaning time.

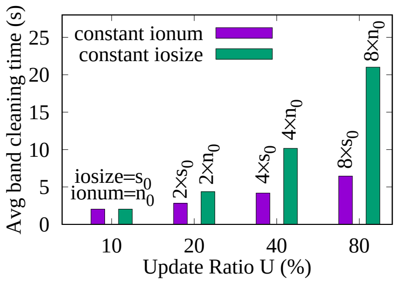


Fig. 5. Compare iosize and ionum.

results in the same highest cleaning time. This is because migrating data into a full zone incurs reading and rewriting to that zone and modifying lower LBA will result in more data to be read out and written back. Such phenomenon implies the drive may adopt some read-modify-write technique that can apply the update by reading and rewriting only parts of the zone data (such as *partial read-modify-write* [24]).

When $U = 100\%$, the full zone cases take more time than the empty zone cases. In fact, identical amount of data is migrated in both cases. It is possibly due to zone metadata loading overhead. This may due to some unnecessary reads or some other reasons that are unclear to us.

4.4.2 I/O request size and I/O request number

From the previous test, we know that the average per zone cleaning time goes up when more data are written to the zone. However, it is unclear whether the cleaning time will be the same if the equivalent amount of non-sequential data are written with different numbers of I/O requests or different I/O sizes. So we design the next test to investigate the impact of I/O request size and I/O request number.

Instead of increasing U by raising I/O request number n, the testing program keeps n constant but increases s to raise U (left bars in Fig. 5). For the baseline, we still carry out another set of tests by keeping s the same but increases n. The results are shown in the right bars in Fig. 5. Note that we start from $U = 20\%$, $s_0 = 32KB$, and $n_0 = 800$ in both cases. The zones are initially empty so that the cleaning overhead is close to the effective data movement cost. It can be seen from the figure, in both cases, the cleaning time increases as U grows. But if we increase I/O request size, the cleaning time grows in a slower pace than the cases where we raise the I/O request number. Therefore, we guess that the number of I/O requests to be cleaned contributes more than the size of the I/O requests does to the cleaning time. We verify this hypothesis using the following test.

In this test, we keep the total amount of data as a constant, but vary the I/O request sizes. In other words, if we cut the I/O request size into half, the total number of I/O requests will be doubled. The test program starts from empty zones and update the first 20% of the zones. The I/O request size ranges from $4KB$ to $256KB$. In Fig. 6, we can see that although the same amount of data is going to be migrated, the cleaning time varies a lot. Also we observe that the cleaning time is nearly inverse proportional to the I/O request size when compared to the theoretical inverse proportional plot in the figure. We conclude that the zone cleaning time is affected more by the I/O request number than the I/O request size. This probably is because the time for random seeks when migrating data blocks and/or the mapping manipulation (read/update) cost of the non-sequentially written data dominates the total cleaning time.

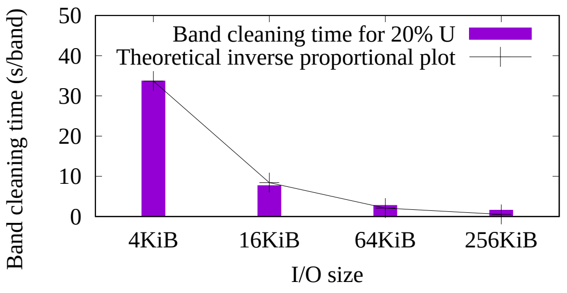


Fig. 6. Average per zone cleaning time of different write request sizes

4.4.3 Zone data distribution within the media cache

The ordering of the non-sequential write requests determines the layout of the data blocks belonging to the same zone in the media cache (FIFO journaling, Sec. 2.5). In this test we investigate the impact of data layout in the media cache on the average per zone cleaning time with other factors discussed above fixed.

The program issues non-sequential 4KB writes to the starting 1% of 1024 zones with two different media cache data layout:

- Clustered: the test program non-sequentially writes the starting 1% of the first zone, then the starting 1% of the second zone, and all the way to the last zone. The data blocks from the same zone is clustered together in the media cache.
- Interleaved: the test program issues a 4KB non-sequential write to the first zone, then issues another 4KB write to the second zone, and all the way to the last zone. Then it starts over again from the first zone in a round-robin fashion until each of the zones has the first 1% updated. Here the data blocks from the same zone are interleaved in the media cache.

We measure the average per zone cleaning time after the writes and surprisingly find that the interleaved writes have a lower average per zone cleaning overhead than that of the clustered writes. From the test we verify that the drive does FIFO journaling and aggressive media cache cleaning (as discovered in Sec. 4.1). When taking a close look, we find that the idle cleaning in the clustered case starts from a non-sequential zone number of 563, not the expected 1024, which indicates that the drive starts the cleaning even earlier than the test finishes. So, we hypothesize that for the interleaved writes the drive performs a similar earlier cleaning and this may be the reason of a lower average per zone idle cleaning time.

In order to verify this hypothesis, we perform the write workload again, and immediately read all the data blocks in the write ordering after the completion of the writes. We trace the latency, the throughput, and the number of non-sequential zones through the test (Fig. 7).

The non-sequential zone number plot in the last row of Fig. 7 proves this hypothesis of early media cache cleaning. In the clustered case, the non-sequential zone number keeps increasing until around 600 seconds when the internal cleaning kicks in. The cleaning process forces the non-sequential zone number to stop increasing and to keep roughly constant until the writes stop. One thing to notice is that media cache cleaning brings down the average throughput. By contrast, in the interleaved case the number of non-sequential zones grows quickly to 1024 because of the round-robin non-sequential writes, then drops twice. This implies the SMR drive cleans the media cache two times before finishing all writes. Accordingly, each time the drive cleans the media cache, the throughput drops and the latency skyrockets.

From the latency and throughput plots of the two cases, we can see clearly the boundary between the write (featuring higher latency, lower throughput, and long duration) and read operations (featuring the opposite). In the clustered case, if we zoom in the read latency plot, there is a noticeable phase change during the read task indicating the first portion of the media cache has already been cleaned out, because the read/write head movement pattern is different from that when accessing the data blocks still residing in media cache. Similarly, there is also a distinct phase change during the read task for the interleaved case. Such phase changes during read requests also confirm our hypothesis that the media cache cleaning has already started before the writes complete for the interleaved case. This actually explains why the interleaved workload needs less cleaning time at the end, because earlier cleaning has already moved some data blocks back to the targeted zones so that less data needs to be migrated for one targeted zone during the idle time. By contrast, the clustered case needs to clean fewer number of zones, but each of the remaining zones has all the new data residing in the media cache. Lastly, we can see that cleaning process starts after a different amount of total data written with different numbers of non-sequential zones. The cleaning durations are also different for different situations. It is unclear when the drive triggers and stops the media cache cleaning.

5 Systems Implications

Here we summarize the systems implications of the performance characteristics of HA-SMR drives to provide a reference point for the HA-SMR systems designers.

Sequential Writes v.s. Non-sequential Writes

The long-held understanding that sequential writes have better performance than non-sequential writes does not apply to HA-SMR drives although the definitions of sequential writes and non-sequential writes are somewhat different from the traditional ones. In some cases, we observe that non-sequential writes could perform better than sequential writes (Fig. 2). Here are several reasons for that:

- Non-sequential writes will be redirected to the media cache which is located in the higher performance outer tracks (Table 3);
- Media cache buffering converts random writes into physically sequential writes;
- Sequential writes could be severely affected by the open zone issue; and
- Light and bursty workloads do not suffer from the performance degradation caused by media cache cleaning as write intensive workloads experiences.

Open Zone Number

Due to the internal limitation of the open zone resource in the HA-SMR drive, a limited number of zones can be open for writing. Frequent switching among sequential written zones means frequent disk synchronizations for the open zone meta-data, and it will then make the write performance drop sharply, even if the application is performing sequential writes. Therefore, system designers should also avoid making the working set for sequential writes beyond the recommended maximum number of open zones.

Non-sequential Zone Number

The penalty of cleaning non-sequential writes is more severe when more zones are non-sequentially written, in which case the low throughput interval is prolonged. During the cleaning time, the throughput drops three orders of magnitude compared with normal performance. Disk vendors may give a recommended maximum number of non-sequential written zones. However, this number is only a rough estimate considering the overall capability of the drive including the size of the media cache, the media cache map size, the disk's internal processing power, the available disk RAM supporting the cleaning, etc. If non-sequential writes are inevitable, the application developers for HA-SMR drives should keep the number of non-sequential zones as few as possible to avoid this severe performance degradation. From another perspective, the HA-SMR drives will fit well in a hierarchical storage architecture as the second tier storage where the first tier – possibly flash – can filter out most of the requests of non-sequential writes. It may also work well as the primary tier if an application/software layer can avoid or reduce non-sequential writes.

Average per Zone Cleaning Time

A good estimation of the zone cleaning time is important for the systems designers to speculate whether and how much internal media cache cleaning will impact the ongoing workloads. Generally, the cleaning time is positively correlated with the total amount of data to be rewritten and the number of buffered write requests to be migrated. Between the two, the number of requests contributes more to the cleaning time. This will give us a hint that larger non-sequential I/O size is favorable in a sense that the cleaning time will be shorter because it results in fewer non-sequential write requests logged in the media cache. Moreover, the drive may trigger cleaning earlier while the drive is still busy.

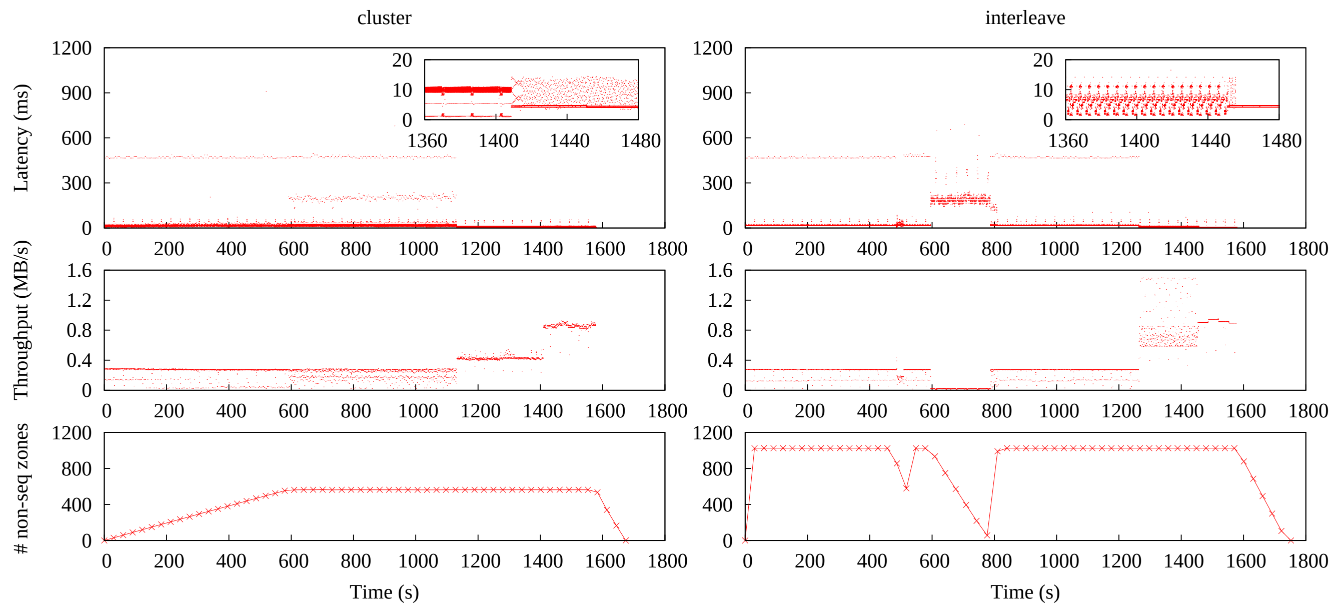


Fig. 7. Zone cleaning time for data in different zone data layout in media cache.

TABLE 4
Zone Specific Analysis for MSR Traces (sorted by the write request #)

trace	total req #	write req #	total write size (GB)	footprint (#zones)*	Write Ratio (%)	trace	total req #	write req #	total write size (GB)	footprint (#zones)*	Write Ratio (%)
wdev_3	682	671	0.003	5	98.39	usr_0	2,237,889	1,333,406	13.077	34	59.58
wdev_1	1,055	1,055	0.005	4	100.00	src2_0	1,557,814	1,381,085	9.340	38	88.66
rsrch_1	13,780	13,738	0.159	84	99.70	web_0	2,029,945	1,423,458	11.673	35	70.12
src2_1	657,774	14,104	0.180	125	2.14	**src1_2**	**1,907,773**	**1,423,694**	**44.147**	**16**	**74.63**
web_3	31,380	21,330	0.424	14	67.97	ts_0	1,801,734	1,485,042	11.340	47	82.42
hm_1	609,311	28,415	0.541	11	4.66	stg_0	2,030,915	1,722,478	15.090	30	84.81
web_2	5,175,368	38,963	0.784	40	0.75	usr_2	10,570,046	1,994,612	26.469	926	18.87
rsrch_2	207,587	71,223	0.289	72	34.31	src1_1	45,746,222	2,170,271	30.349	381	4.74
web_1	160,891	73,833	0.649	40	45.89	proj_1	23,639,742	2,496,935	25.576	1,475	10.56
proj_4	**6,465,639**	**95,865**	**1.010**	**294**	**1.48**	hm_0	3,993,316	2,575,568	20.477	45	64.50
proj_3	2,244,644	116,341	2.627	254	5.18	**prn_1**	**11,233,411**	**2,769,610**	**30.785**	**1,222**	**24.66**
mds_1	1,637,711	116,676	1.540	188	7.12	proj_2	29,266,482	3,624,878	168.686	1,474	12.39
wdev_2	181,266	181,077	1.407	32	99.90	proj_0	4,224,524	3,697,143	144.267	34	87.52
stg_1	**2,196,861**	**796,452**	**5.986**	**224**	**36.25**	usr_1	45,283,980	3,857,714	56.127	1,886	8.52
src2_2	1,156,885	805,955	39.282	275	69.67	prn_0	5,585,886	4,983,406	45.965	85	89.21
wdev_0	1,143,261	913,732	7.146	27	79.92	prxy_0	12,518,968	12,135,444	53.797	58	96.94
mds_0	1,211,034	1,067,061	7.365	19	88.11	src1_0	37,415,613	16,302,998	809.159	478	43.57
rsrch_0	1,433,655	1,300,030	10.818	59	90.68	prxy_1	168,638,964	58,224,504	724.816	169	34.53

* Here footprint is defined as the number of zones that are written by the traces.

6 Replay of Microsoft Research Cambridge Traces

Previous tests mainly focus on investigating the drive's zone-specific performance using synthesized traces. We are also interested in the I/O performance of HA-SMR drives under typical real-life workloads. Besides, we can apply the knowledge obtained from the previous performance tests (Sec. 5) to help predicting and analyzing the real-world trace replay results.

We replay the 36 one-week-long Microsoft Research Cambridge (MSR) traces [25] which are collected from enterprise data center workloads. Traces are replayed against both the HA-SMR drives and baseline HDDs. The HDD model is Seagate ST6000NM0034 (6TB SAS HDD, 7.2K RPM). We replay the MSR trace using fio with the *sync* engine. Write cache and read-ahead are turned on for both HDD and SMR drives to mimic realistic production environment. This is different from the previous settings of tests. Moreover, as reading from a location beyond the write pointer will return synthesized fill data from the SMR drive's RAM and result in unrealistic high throughput (Sec. 2.5), we "warm up" the SMR drive before replaying by making the zones full such that the read operations will get fill data from the disk. Our current tests simply "blast" all the I/O requests to the drive without referring to the time stamps of I/O operations.

6.1 Zone-Specific Analysis for MSR Traces

Before the trace replay, the following key statistics are analyzed for these traces: total number of operations, number of write operations, total write size, zone footprint of the writes, ratio of write operations (Table 4).

Those statistics are selected because the following reasons. The number of write operations and the total write amount have direct impact on the blocking media cache cleaning (Sec. 2.7). As the write operations of legacy zone-unaware workloads are almost non-sequential (mostly

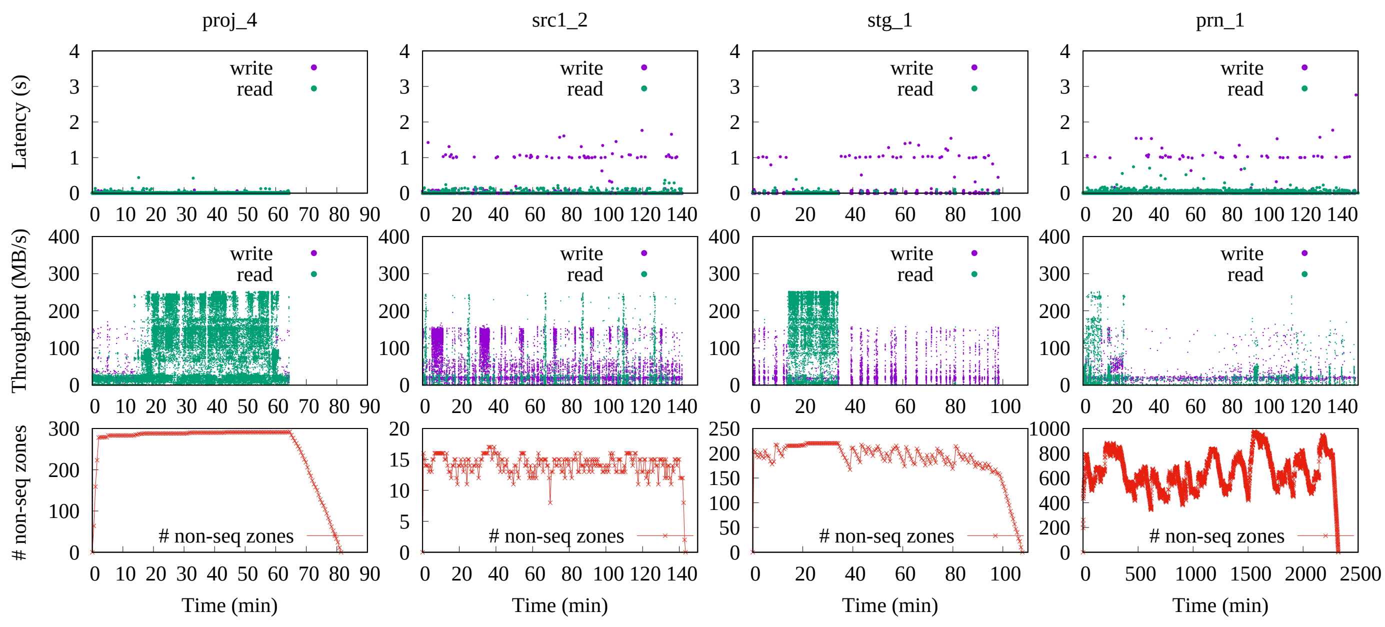


Fig. 8. Replay results of selected traces.

never happening at the write pointers), the number of write operations can be regarded as the media cache mapping entries required by the trace. Similarly, the total write amount is a good estimate for the media cache capacity requested by the trace. If the requested resources exceed the drive's media cache capacity (Table 3), cleaning will become blocking which affecting both read and write performance (Sec. 2.7). The total number of operations and the write operation ratio reflect how intensive data is injecting write requests into the SMR drive. Zone footprint for write operations will affect the media cache cleaning efficiency. The more zones the non-sequential writes span, the less efficient the cleaning process is (Sec. 4.4).

From the statistics, we find that it is the media cache mapping size, rather than the media cache size that becomes the bottleneck and causes blocking media cache cleaning. While the replay results of all traces are conducted, four representative ones are especially studied who represent different media cleaning severity, i.e., `proj_4`, `stg_1`, `src1_2`, and `prn_1` (high-lighted in bold in Table 4).

6.2 Representative Traces Analysis

Trace `proj_4` has a small number of write operations and the total write amount is small too, so we hypothesize that it will not trigger blocking media cache cleaning even the write operations span as many as 294 zones. By contrast, the other three selected traces are expected to trigger blocking media cache cleaning as they have more non-sequential writes than the media cache mapping table size. According to Sec. 5, we hypothesize that media cache cleaning caused performance degradation will be positively related to the zone footprint. So we expect the severity of media cache cleaning impact to be `proj_4` < `src1_2` < `stg_1` < `prn_1`.

Fig. 8 shows the latency, bandwidth and the non-sequential zone plots of the selected traces. It can be observed from the non-sequential zone number plot that `proj_4` never triggers media cache cleaning, while the others do. Note that even having a total write amount as small as 5.99GB, `stg_1` still triggers cleaning.

We can see from the performance plots of `stg_1` that each time the drive starts media cache cleaning, the throughput decreases and the latency goes high. During the almost read-only intervals, the non-sequential zone number keeps constant. This is because there are fewer write operations injecting into the media cache and the drive is still busy such that the idle cleaning never gets a chance to happen.

Comparing `stg_1` and `src1_2`, we find that `src1_2` spans much less number of zones (16) than that of `stg_1` (224). Therefore, it will have less performance degradation even it imposes more demand for the media cache capacity and mapping table. `prn_1` has the most zone footprint (1222), implying a serious media cache cleaning impact.

The average write latencies of `proj_4`, `stg_1`, `src1_2`, and `prn_1` are $712.55\mu s$, $4335.34\mu s$, $5708.74\mu s$, and $28426.5\mu s$ respectively indicating an increasing degree of performance degradation.

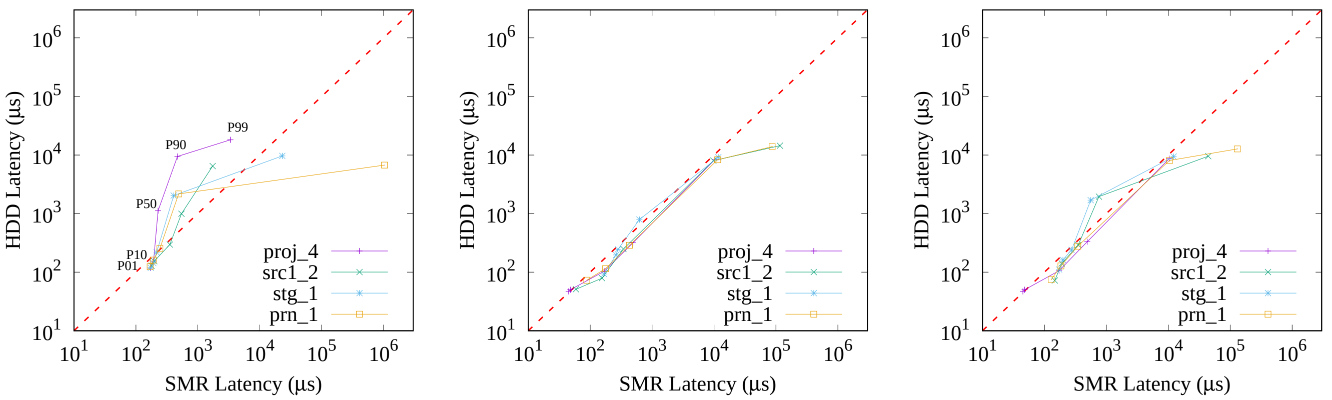
One thing to notice is that the read latency is not affected as much as the write latency. For example, the average read latencies for `proj_4`, `stg_1`, `src1_2`, and `prn_1` are $592.11\mu s$, $4767.97\mu s$, $962.704\mu s$, and $6779.09\mu s$ respectively. The read latency of `stg_1` is much less than that of `src1_2`. One possible reason is that the read-ahead mechanism of the disk will reduce the latency of subsequent read requests if the read requests are located close to each other. In `stg_1`, 93.5% of the read requests have a sub-millisecond latency because of this read ahead benefit. Although media cache cleaning does not directly affect the read performance, it may indirectly reduce the read ahead benefit by "relocating" the modified data blocks to media cache and breaking the physical consecutiveness of data accesses.

6.3 Latency Analysis

We summarize the average latency of all the traces in Table. 5. We also collect the 1st, 10th, 50th(median), 90th,

TABLE 5
MSR Traces Average Latency (sorted by the write request #, see Table 4)

Trace	HDD Avg Lat.(μs)	SMR Avg Lat. (μs)	Trace	HDD Avg Lat.(μs)	SMR Avg Lat. (μs)	Trace	HDD Avg Lat.(μs)	SMR Avg Lat. (μs)	Trace	HDD Avg Lat.(μs)	SMR Avg Lat. (μs)
wdev_3	240.92	221.93	**proj_4**	**451.27**	**593.90**	usr_0	1267.99	2300.85	hm_0	2376.67	3213.04
wdev_1	159.72	220.67	proj_3	384.55	853.91	src2_0	612.04	1461.26	**prn_1**	**2206.03**	**12116.30**
rsrch_1	2207.60	8579.06	mds_1	781.69	1547.47	web_0	2071.97	2489.80	proj_2	1191.22	2248.36
src2_1	743.30	977.02	wdev_2	490.66	718.95	**src1_2**	**978.80**	**4445.12**	proj_0	876.53	1480.60
web_3	2400.37	1555.28	**stg_1**	**848.13**	**2683.33**	ts_0	536.96	1068.41	usr_1	1415.70	4703.44
hm_1	436.98	785.13	src2_2	2836.14	3662.42	stg_0	843.51	2588.73	prn_0	2013.20	4347.09
web_2	344.52	406.65	wdev_0	442.32	920.41	usr_2	1781.26	4209.63	prxy_0	1025.67	592.48
rsrch_2	1321.52	951.67	mds_0	629.77	827.67	src1_1	729.98	1376.58	src1_0	1490.85	3007.77
web_1	1478.16	1050.01	rsrch_0	423.46	2501.85	proj_1	1471.86	4220.74	prxy_1	2943.32	3259.39



(a) Write requests (b) Read requests (c) Write and read requests

Fig. 9. Sampled Q-Q plot between HDD and SMR drive percentile latency numbers for selected MSR traces. (Note that the plots are in double log scale).

and 99th percentile latency numbers, and plot the percentile latency of HDD drives against those of the SMR drives, i.e. a sampled Q-Q plot. In Fig. 9, we show the Q-Q plot of the same set of representative traces as Sec. 6.2. The most bottom left dot represents the 1st percentile (P01), the second means the 10th latency (P10), so on and so forth. The last point (most top right) represents the 99th percentile latency (P99). The dashed line across the figure shows the boundary where the SMR drive latency is equal to the HDD latency such that any dots to the top left of the dashed line means the SMR drive has a smaller percentile latency, and vice versa.

The Q-Q plot of write requests (Fig. 9a) has greater deviation from the diagonal dashed line than that of the read requests (Fig. 9b). The Q-Q plot for both the I/O requests (Fig. 9c) is a weighted combination of that of the write and read requests.

For write operations (Fig. 9a), traces have comparable 1st and 10th percentile latencies between HDD and SMR drives, and SMR drives have a better 50th and 90th percentile latencies. This is due to the media cache buffering effect which converts non-sequential writes into physically sequential ones such that most the SMR write operations have a reduced write latency. However, for some 99th percentile write latencies of SMR drives, they go up by over 2 order of magnitudes from the 90th and this phenomenon does not exists in HDD. It means 1 out of every 100 write operations may experience a latency in seconds. This is the price to pay when cleaning the media cache. This implies legacy application will experience more serious performance jitters in write operations. Nevertheless, the 99th SMR write percentile latencies of other traces stay above the dashed line indicating a consistent advantage of HA-SMR drives over HDDs. Such traces never trigger blocking cleaning (e.g. `proj_4`) or only have a moderate blocking cleaning happening (e.g. `stg_1`) as its non-sequential writes span over a smaller number of zones.

For read operations, SMR drives has similar 10th, 50th and 90th latency distribution to that of HDD as the shingled constraint has less impact on read operations. Some traces, e.g. `prn_0`, `prn_1` (show in Fig. 9b), `proj_0`, `proj_1`, `usr_0`, `usr_1`, `usr_2`, experience a longer 99th read latency. We believe the reason could also possibly be the media cache cleaning as they coincide with the long 99th percentile write latency of the same traces. It may also because the non-sequential write requests reduce the read-ahead benefits.

6.4 Discussion

Note that the baseline HDD has a slightly higher RPM than that of SMR drives so the comparisons will have some bias in favor of HDD. Moreover, in a realistic setting, the I/O commands will come to the drive intermittently such that there may be idle time for the drive to clean the media cache without affecting the performance.

7 H-BUFFER

The special performance characteristics especially the performance degradation observed in Sec.4.2 and Sec. 4.3 raise quite a challenge for software developers using the HA-SMR drives in a large storage system. To confront

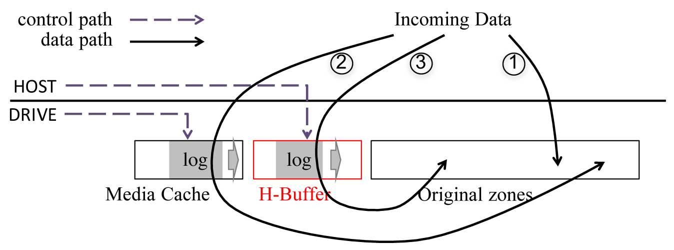


Fig. 10. H-Buffer and the three data paths for HA-SMR drive.

this challenge, by exploiting the unique HA-SMR hardware features and host interacting model, we propose a *Host-controlled indirection Buffer* (H-Buffer). H-Buffer embodies an HA-SMR three-data-path system (the *mechanism*) (Fig. 10) which can support a broad spectrum of workload switching algorithms (the *policies*). We believe such separation between mechanism and policy can potentially lead to various remedies to the performance degradation problem due to undesirable workloads.

7.1 H-Buffer Description

A small number of the zones either shingled or conventional with a user configurable size (e.g. comparable to the size of media cache) of the drive are reserved as H-Buffer to which incoming data can be redirected by the host. An LBA-PBA mapping table that corresponds to the redirection is maintained in host memory while data blocks are logged as a self-describing journal similar to the media cache for mapping recovery. Besides, the host (not the drive) is responsible for migrating buffered data back to the targeted zones (H-Buffer cleaning). Practically, this H-Buffer con be considered as a host-controlled media cache.

7.2 Three Data Paths for HA-SMR Drives

As shown in Fig. 10, an HA-SMR drive natively supports data path ① and ②. ①, the *direct data path*, represents sequential data being directly written to shingled zones; and ②, the *media cache data path*, denotes non-sequential data being buffered into the media cache and later migrated to the intended zones (Sec. 2.5). Our H-Buffer completes the picture by creating a new data path (③, the *H-Buffer data path*), which is similar to the media cache data path but is controlled by the host instead of the drive. Note that Host Aware is the only SMR model that is able to support all three data paths because it is the only one that can both handle non-sequential writes and provide zone information to enable accurate host control.

The three data paths each has its own strength: 1) The direct path can accept sequential data without any indirection/migration overhead. 2) Media cache data path can handle non-sequential data cleaning without transferring data back and forth with the host, thus has a higher data migrating bandwidth and consumes no extra computational resource from the host. 3) H-Buffer data path can support enhanced cleaning algorithm leveraging the host's bigger memory and more powerful processors such that it can be expected to do a better job on data cleaning. In addition, H-Buffer has the ability to redirect sequential writes too.

7.3 Data Management Separation between The Host and The Drive

Data path ② has the managing logic in the drive, while ③ has the controlling intelligence within the host. This three-data-path model separates the handling of non-sequential writes and even sequential writes between the host and the drive. Besides, the policy engine switching data requests among the three data paths is also part of the host controlling logic. This enables the system designer to find an optimized combination between the host management and the drive management.

A host-side management can alleviate the media cache cleaning overhead by sharing the data handling task. Actually the host may potentially do a better job because it has more computational power and RAM size so that it can support more sophisticated buffering and cleaning algorithms. In addition, a host-side management has the advantages of having the knowledge of the workload characteristics. On the other hand, drive-side management does not consume host computational resources and can leverage the disk internal bandwidth to do high speed non-sequential data migration. In all, H-Buffer and the three-data-path system essentially open a big design space for the policies switching the requests among the three data paths and the separation between the host and drive management in order to combine all the advantages under various workloads.

8 Case Study: Solving Open Zone Issue with H-Buffer

In this section, we try to demonstrate the potential of H-buffer by investigating how it can be used in dealing with one of the unique features of HA-SMR drive. That is, the open zone issue for sequential write zones. We find that there is a clear performance drop if the number of open zones exceeds the recommend maximum number (Sec. 4.2).

Applications should always respect this recommendation of 128 maximum number of sequential zones if they do not want to hurt the performance by the open zone thrashing phenomenon. However, if some applications do have a requirement to sequentially write over more than 128 zones simultaneously, the H-Buffer can be used to absorb the incoming writes and later migrate the data to the intended zones.

We demonstrate H-Buffer's potential by designing a simple workload switching policy (named: *open-zone-policy*) which addresses the open zone issue (Sec. 4.2). The open-zone-policy detects concurrent sequential write streams (each stream corresponds to sequential writes to a single open zone) from the workload. If there are more streams than the recommended maximum number of open zones, the policy will switch the sequential write streams into the H-Buffer to maintain the number of sequential open zones no more than 128.

To evaluate the open-zone-policy, we create a micro-benchmark workload that contains 200 concurrent sequential write streams. We assume that system reserves 100 zones for H-Buffer. We compare the total run time with the open-zone-policy switched on and off:

- policy switched off: 1000 write operations are directly issued to 200 zones.
- policy switched on: 1000 write operations are initially redirected to the H-Buffer (25.6G, comprised of 100 zones and comparable to the size of the media cache). Then the host performs the H-Buffer cleaning: read

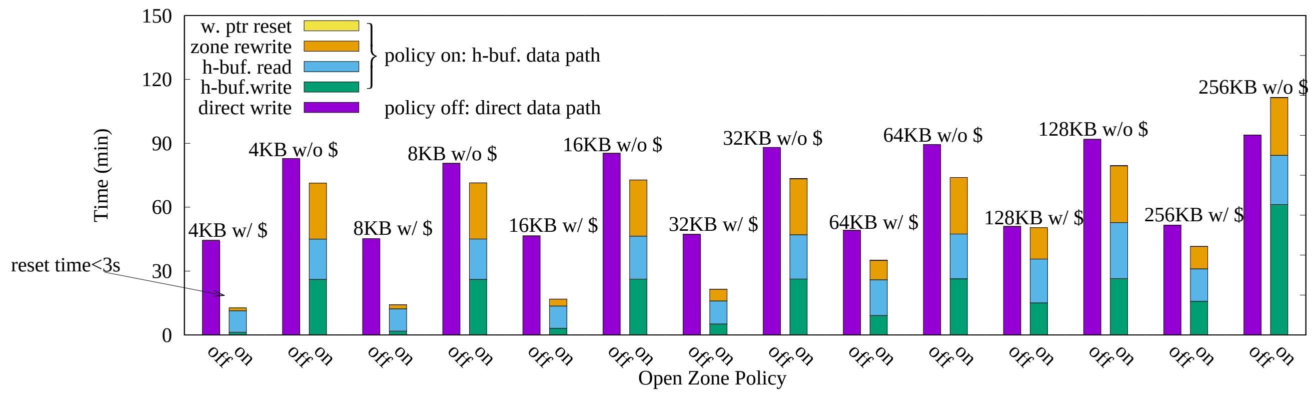


Fig. 11. H-Buffer helps addressing the open zone issue. ("w/ $" means with the write-cache and the read-ahead enabled, and vice versa.)

data out from the H-Buffer, write them to the intended 200 zones, and finally reset the write pointer for the 100 H-Buffer zones. Note that this is not the normal case of using H-buffer because typically not all the write operations need to be redirected.

We perform the test with the I/O size ranging from 4KB to 256KB. Besides, for each size we experiment with the drive's write cache and read ahead enabled or disabled. We compare the total run time with open-zone-policy turned on and off (Fig. 11). When open-zone-policy is on, the time is broken down into H-Buffer redirected write, cleaning read, zone rewrite and write pointers reset.

We find that resetting 100 zones only takes at most 3 seconds which is negligible compared to the data transmission time in the figure. In small I/O sizes, the H-Buffer data path has shorter total time than that of the direct write path. As the I/O size increases, the total time without open-zone-policy grows slowly and gets surpassed by the total time with the open-zone-policy. Also, we see a greater advantage for the open-zone-policy when the write-cache and read-ahead functions are enabled for the drive. Therefore, the H-Buffer will bring performance benefit and the benefit is more significant when write cache and read ahead are enabled and the I/O request sizes are smaller.

In conclusion, even though we do extra I/O operations in the H-Buffer case, it still takes less total time than that of directly writing to a large number of zones. In this case study, we simply switch all the I/O requests to H-Buffer data path. However, practical policy can be more sophisticated by handling each individual I/O request differently and switching each write operation into an optimal data path.

9 Related Work

9.1 SMR Drive Characterization/Evaluation

Black et al. [26] studied the particular SMR characteristics in large-scale cold-data archival storage systems in which the rack-level power provision requires the SMR drives to be frequently spun up and down. They investigated both performance and reliability characteristics for HA-SMR and DM-SMR such as spin up latency, disk failure, etc. They deploy HA-SMR drives as DM-SMR drives without utilizing or evaluating the zoned block APIs while our work emphasizes the unique HA-SMR zone-aware features and its system performance implications. Besides, our work focuses on microscopic performance characteristics for the HA-SMR drive while Black et al. [26] studied the statistic behavior for the SMR drives from a macroscopic perspective.

Aghayev and Desnoyers [12] use both software and hardware approaches for DM-SMR drive reverse engineering. In the software part, they use *fio* and measure the latency of the I/O to determine the internal structure of the drive. The hardware part is a high-speed camera monitoring the movement of the read/write head of the drive through a window on the drive. Different from Skylight's hardware and software approach, we leverage the richer-featured zoned block APIs to collect more information from inside the drive to interpret the performance results and to understand the internal structure of the drive.

9.2 SMR Data Handling Solutions

Amer et al. explored the design space for the data layout of SMR drives [27] [28]. They proposed to have multiple bands of tracks separated by guard tracks to avoid interference between each other. Cassuto et al. [15] provided a definition of *indirection system* for SMR drives which is a collection of data structures and algorithms that map Logical Block Addresses (LBAs) to Physical Block Addresses (PBAs). Besides, they designed two indirection architectures for the indirection system, namely shingled set-associative disk cache and circular buffers with S-blocks.

H-SWD [29] [30] adopted a similar circular log layout and also implemented hot/cold data identification for data placement decision. Hall et al. [16] divided SMR drives into two shingled write regions, i.e. *I-region* (larger) and *E-region* (smaller). I-Region means *indirection region*, which serves as the native space for data. In I-Region, data is written in a sequential manner and the corresponding LBA-PBA translation is done by the indirection system. He and Du exploited static [17] and dynamic [18] track level mapping schemes to reduce the write amplification and garbage collection overhead.

SFS [31] uses the full knowledge of the disk layouts and allows more efficient optimization compared to a disk firmware based approach. Jin et al. proposed a file system called HiSMRfs [32] which separates the meta-data and data space into SSD and SMR drives.

Manzanares et al. designed a zone-based extent allocator (ZEA) [33] that only performs sequential allocation for data and meta-data on SMR drives to maintain the shingled constraint. ZEA does not consider exploiting the non-sequential data handling path that exists in HA-SMR drives.

Our work differs from other SMR data handling solutions in that we focus on the software solution for HA-SMR model. We propose the three-data-path system (enabled by H-Buffer) for HA-SMR drives by exploiting each of their unique interacting model.

10 Conclusions and Future Work

We present a study on Host-Aware SMR drives by carrying out in-depth performance evaluations on several HA-SMR sample drives. Specifically, we investigate the drive internals, the performance impact of the zoned block APIs and the factors that affect the zone cleaning efficiency. Based on the empirical observations, we summarized the system impact and utilize the knowledge obtained from the evaluation to analyze the replay results of real world traces. Besides, we propose a host-controlled indirection buffer framework for improving the performance on HA-SMR drive systems. A case study of the open zone issue shows the potential of such host-controlled buffer.

In the future, we plan to further investigate the media cache cleaning mechanism and explore more production workloads for testing. Moreover, we are going to comprehensively investigate the data path switching policies when using H-buffer to fit various workloads such that HA-SMR drives can be used to construct large scale storage systems that support various types of workloads.

Acknowledgments

This work has been supported by NSF I/UCRC Center for Research in Intelligent Storage (CRIS) and the National Science Foundation (NSF) under awards 130523, 1439622, and 1525617.

References

[1] S. Piramanayagam, “Perpendicular recording media for hard disk drives,” *Journal of Applied Physics*, vol. 102, no. 1, p. 011301, 2007.

[2] R. E. Rottmayer, S. Batra, D. Buechel, W. Challener, J. Hohlfeld, Y. Kubota, L. Li, B. Lu, C. Mihalcea, K. Mountfield *et al.*, “Heat-assisted magnetic recording,” *Magnetics, IEEE Transactions on*, vol. 42, no. 10, pp. 2417–2421, 2006.

[3] M. H. Kryder, E. C. Gage, T. W. McDaniel, W. Challener, R. E. Rottmayer, G. Ju, Y.-T. Hsia, M. F. Erden *et al.*, “Heat assisted magnetic recording,” *Proceedings of the IEEE*, vol. 96, no. 11, pp. 1810–1835, 2008.

[4] R. L. White, R. Newt, and R. F. W. Pease, “Patterned media: a viable route to 50 gbit/in 2 and up for magnetic recording?” *IEEE Transactions on Magnetics*, vol. 33, no. 1, pp. 990–995, 1997.

[5] R. Wood, M. Williams, A. Kavcic, and J. Miles, “The feasibility of magnetic recording at 10 terabits per square inch on conventional media,” *Magnetics, IEEE Transactions on*, vol. 45, no. 2, pp. 917–923, 2009.

[6] I. Tagawa and M. Williams, “High density data-storage using shinglewrite,” in *Proceedings of the IEEE International Magnetics Conference*, 2009.

[7] G. Gibson and M. Polte, “Directions for shingled-write and twodimensional magnetic recording system architectures: Synergies with solid-state disks,” *Parallel Data Lab, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-PDL-09-014*, 2009.

[8] Seagate Technology, “Affordable active archive hard drives for cloud storage.” http://www.seagate.com/enterprise-storage/hard-disk-drives/archive-hdd/.

[9] INCITS T10 Technical Committee, “Information technology - zoned block commands (zbc).” Draft Standard T10/BSR INCITS 536, American National Standard Institute, Inc., December 2015. [Online]. Available: http://www.t10.org/drafts.htm

[10] INCITS T13 Technical Committee, “Zoned-device ata command set (zac) working draft,” http://www.t13.org/Documents/MinutesDefault.aspx?DocumentType=4&DocumentStage=2, accessed: 2016-01-23.

[11] T. Feldman and G. Gibson, “Shingled magnetic recording areal density increase requires new data management,” *USENIX ;login:*, vol. 38, no. 3, 2013.

[12] A. Aghayev and P. Desnoyers, “Skylight–a window on shingled disk operation,” in *13th USENIX Conference on File and Storage Technologies (FAST15)*, 2015, pp. 135–149.

[13] T. R. Feldman, “Host aware smr.” http://open-zfs.org/w/images/2/2a/Host-Aware_SMR-Tim_Feldman.pdf, OpenZFS Develop Summit, San Francisco, November 2014.

[14] R. Wood, “The feasibility of magnetic recording at 1 terabit per square inch,” *Magnetics, IEEE Transactions on*, vol. 36, no. 1, pp. 36–42, 2000.

[15] Y. Cassuto, M. A. Sanvido, C. Guyot, D. R. Hall, and Z. Z. Bandic, “Indirection systems for shingled-recording disk drives,” in *26th IEEE Symposium on Mass Storage Systems and Technologies (MSST10)*. IEEE, 2010, pp. 1–14.

[16] D. Hall, J. H. Marcos, and J. D. Coker, “Data handling algorithms for autonomous shingled magnetic recording hdds,” *Magnetics, IEEE Transactions on*, vol. 48, no. 5, pp. 1777–1781, 2012.

[17] W. He and D. H. Du, “Novel address mappings for shingled write disks,” in *6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14)*, 2014.

[18] W. He and D. H. Hu, “Smart: An approach to shingled magnetic recording translation,” in *15th USENIX Conference on File and Storage Technologies (FAST17)*, 2017.

[19] C. Li, P. Shilane, F. Douglis, D. Sawyer, and H. Shim, “Assert (! defined (sequential i/o)),” in *6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14)*, 2014.

[20] T. R. Feldman, Personal communication, January 2016.

[21] HGST, “libzbc,” https://github.com/hgst/libzbc.

[22] INCITS T10 Technical Committee, “Scsi block commands - 4 (sbc-4),” http://www.t10.org/members/w_sbc4.htm.

[23] J. Axboe, “Flexible i/o tester,” https://github.com/axboe/fio.

[24] S. Poudyal, “Partial write system,” Jun. 30 2015, uS Patent 9,070,378.

[25] D. Narayanan, A. Donnelly, and A. Rowstron, “Write off-loading: Practical power management for enterprise storage,” *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, p. 10, 2008.

[26] R. Black, A. Donnelly, D. Harper, A. Ogus, and A. Rowstron, “Feeding the pelican: Using archival hard drives for cold storage racks,” in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[27] A. Amer, D. D. Long, E. L. Miller, J.-F. Paris, and S. Schwarz, “Design issues for a shingled write disk system,” in *26th IEEE Symposium on Mass Storage Systems and Technologies (MSST10)*. IEEE, 2010, pp. 1–12.

[28] A. Amer, J. Holliday, D. D. Long, E. L. Miller, J. Paris, and T. Schwarz, “Data management and layout for shingled magnetic recording,” *Magnetics, IEEE Transactions on*, vol. 47, no. 10, pp. 3691–3697, 2011.

[29] C.-I. Lin, D. Park, W. He, and D. H. Du, “H-swd: Incorporating hot data identification into shingled write disks,” in *20th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS12)*, 2012.

[30] D. Park, C.-I. Lin, and D. H. Du, “H-swd: A novel shingled write disk scheme based on hot and cold data identification,” in *10th USENIX Conference on File and Storage Technologies (FAST12)*, 2012.

[31] D. Le Moal, Z. Bandic, and C. Guyot, “Shingled file system host-side management of shingled magnetic recording disks,” in *IEEE International Conference on Consumer Electronics (ICCE12)*. IEEE, 2012, pp. 425–426.

[32] C. Jin, W.-Y. Xi, Z.-Y. Ching, F. Huo, and C.-T. Lim, “Hismrfs: A high performance file system for shingled storage array,” in *30th International Symposium on Mass Storage Systems and Technologies (MSST14)*, June 2014, pp. 1–6.

[33] A. Manzanares, N. Watkins, C. Guyot, D. LeMoal, C. Maltzahn, and Z. Bandic, “Zea, a data management approach for smr,” in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.