Data Management Design for Interlaced Magnetic Recording

Fenggang Wu Baoquan Zhang Zhichao Cao Hao Wen Bingzhe Li
Jim Diehl Guohua Wang[†] David H.C. Du
University of Minnesota, Twin Cities [†]South China University of Technology

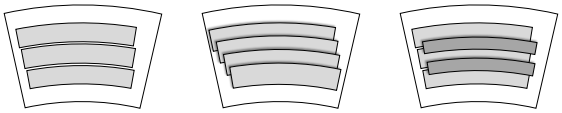
Abstract

Interlaced Magnetic Recording (IMR) is a promising technology which achieves higher data density and lower write amplification than Shingled Magnetic Recording (SMR) when used with Heat-Assisted Magnetic Recording (HAMR). In IMR, top (narrower) tracks and bottom (wider) tracks are interlaced so that each bottom track is partially overlapped with two adjacent top tracks. Top tracks can be updated without any write amplification, but updating a data block in a bottom track requires reading and rewriting of the affected data on the two neighboring top tracks if they contain valid data. We investigate efficient data management schemes for IMR in this paper. First, we design a Three-Phase data management algorithm that allocates disk space in three stages according to disk usage. We further propose two techniques, Top-Buffer and Block-Swap, which can be used in IMR to improve the performance of the Three-Phase algorithm. Top-Buffer opportunistically makes use of unallocated top track space as a buffer for updates to the bottom tracks, while Block-Swap progressively swaps hot data in bottom tracks with cold data in top tracks. Finally, we propose our Data Management design for IMR, or DM-IMR, by integrating Top-Buffer and Block-Swap with the Three-Phase scheme. Evaluations with Microsoft Research Cambridge traces show that DM-IMR can increase the throughput and reduce the write amplification for all traces when compared with the Three-Phase baseline scheme.

1 Introduction

The rapid growth of digital content from the cloud, mobile computing, social media, big data, and other emerging applications calls for low cost, but large capacity storage systems [1]. Energy-assisted technologies such as Heat-Assisted Magnetic Recording (HAMR) [2, 3] and Microwave-Assisted Magnetic Recording (MAMR) [4, 5] enable further growth of the areal data density of hard disk drives.

Recently, a promising track layout, namely Interlaced Magnetic Recording (IMR), has been proposed [6,7] and tested in HAMR systems [8, 9] where it accomplishes higher areal density than Conventional Magnetic Recording (CMR, Fig. 1a) while having much less rewrite overhead and potentially higher data density than Shingled Magnetic Recording (SMR, Fig. 1b) [10–12]. MAMR drives are also expected to use IMR.



(a) CMR (b) SMR (c) IMR

Figure 1: Track layout for CMR, SMR, and IMR.

In heat-assisted IMR, as shown in Fig. 1c, track layout is in an interlaced fashion with alternating bottom tracks (lighter color) and top tracks (darker color). Compared with top tracks, bottom tracks are wider and written with higher laser power. As a result, bottom tracks have a greater linear density and data rate than top tracks (each about 27% higher) [8]. Compared to HAMR-SMR, HAMR-IMR potentially increases areal density but significantly reduces rewrite overhead [6, 8, 13].

In IMR, a narrower top track is written on top of the boundary of two adjacent bottom (wider) tracks. In other words, each bottom track is overlapped with two neighboring top tracks. Thus, top tracks can be updated without penalty, but updating a bottom track may require rewriting the two affected top tracks (*rewrite penalty* or *write amplification*). If the top tracks do not contain any valid data, no rewrites are required. Therefore, the performance of IMR depends on its space utilization and data layout design. If in-place updates are used, in the worst case, an update to data in a bottom track may require two reads and three writes.

A three-phase data allocation scheme is proposed by Gao et al. [7, 14] which allocates disk space based on three phases of space usage. In the first phase, if the usage is less than the total capacity of the bottom tracks ($0 \sim 56\%$ usage), all the data is assigned to the bottom tracks sequentially. In the second phase, space will be allocated from every other top track until half of the to-

tal top track capacity is used (56% $\sim$ 78% usage). In the third phase, the remaining top tracks will be used (78% $\sim$ 100% usage). There is no penalty when in-place updating a bottom track during the first phase. During the second utilization phase, in-place updates to bottom tracks will require one rewrite in one of the adjacent top tracks (or no penalty if neither of the two neighboring top tracks has affected valid data). Similarly, bottom updates in the third phase will require one or two top rewrites. We refer to a bottom track that has no valid data on its two adjacent top tracks as a *free* bottom track and one that has valid data on its adjacent top tracks as a *non-free* bottom track.

In [7], the data allocation in all three phases is from outer diameter (OD) tracks to inner diameter (ID) tracks. This pattern may harm data locality by physically separating adjacent data (e.g., the ending track of the first phase and the beginning track of the second phase). To improve this scheme, we propose our *Three-Phase* baseline design which reverses the second phase allocation direction (making it inner tracks to outer tracks) to preserve data locality between phases (Fig. 2).

To further improve the baseline design and make the data management adapt to both the capacity usage and workload hot spots, we propose *DM-IMR*, a Data Management design for IMR. DM-IMR enhances the baseline by using two key techniques: *top track buffering* (Top-Buffer) and *block swapping* (Block-Swap).

Top-Buffer opportunistically takes advantage of the top tracks that have not yet been allocated and uses them to buffer updates to the non-free bottom tracks. Block-Swap can progressively swap hot (frequently updated) bottom track blocks with cold (infrequently updated) top track blocks to reduce the update overhead.

In DM-IMR, Top-Buffer is used in the second phase and most of the third phase with the last few unallocated top tracks serving as the buffer region. Near the end of the third phase, as the disk usage increases and the buffer region begins giving space to user data, Block-Swap will gradually be brought in. When usage is 100%, only Block-Swap is operational.

We implement DM-IMR as well as the baseline schemes in an IMR simulator. Evaluation results show that DM-IMR is able to increase throughput by 7.78$\times$ and can reduce write amplification by 62.8% compared to the baseline for some write intensive workloads.

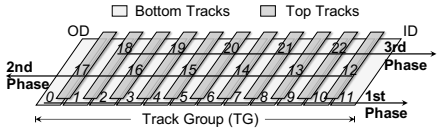


Figure 2: Three-Phase baseline design. Numbers show the addressing and allocation order of the tracks. Note that the second phase has reversed allocation direction for better data locality.

2 Design

We investigate data management methods to address the write amplification issue of IMR (Sec. 1) by proposing two techniques in addition to the Three-Phase baseline: namely top track buffering (Top-Buffer) and block swapping (Block-Swap). We enhance Three-Phase by adding Top-Buffer and Block-Swap in our proposed Data Management design for IMR, called DM-IMR. DM-IMR adapts to the changes in both space utilization and hot spots of the workloads by switching smoothly from Three-Phase, to Top-Buffer, to a combination of Top-Buffer and Block-Swap, and eventually to Block-Swap-only as the usage increases.

2.1 Assumptions

We assume the space manager (e.g., file system, logical volume manager, I/O controller, etc.) allocates an interlaced set of consecutive physical top and bottom tracks, named a *Track Group* or *TG* (Fig. 2), to the applications. How the space manager allocates the space is beyond the scope of this paper, as we only focus on the data allocation and management within one TG.

Although there may be different track capacities within one TG in real production disks due to factors such as zone bit recording and sector defects, we assume the top tracks within one TG have identical capacity and the bottom tracks also have a unified, but higher, capacity. The IMR configurations that we use are based on testing data by Granz et al. [8] and summarized in Table 1. Our data management principles can be easily adapted to other production settings.

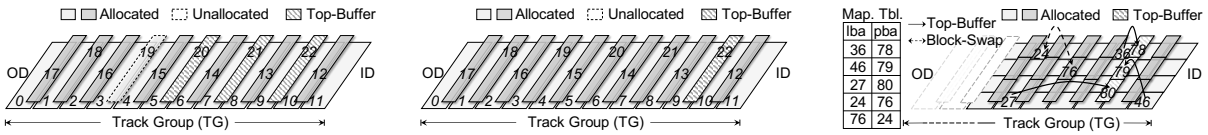
2.2 Top Track Buffering (Top-Buffer)

Fig. 3a shows the design of Top-Buffer in which the last few unallocated top tracks are organized as the Top-Buffer region. Top-Buffer utilizes these unallocated top tracks as buffers for updates to non-free bottom tracks when the TG usage is not 100%. Top-Buffer is able to accumulate multiple updates to the same bottom data block to reduce the rewrite penalty.

When the Top-Buffer is full, we evict buffered blocks to reclaim space using a Sequential Cleaning Policy (SCP). SCP cleans a whole Top-Buffer track at a time by sequentially reading the buffered blocks on the target track (*victim blocks*) and writing them back to their

Table 1: IMR disk configuration.

Basic Parameters	
Median Track pitch	820KTPI
Median top track density	1640KBPI
Median bottom track density	2030 KBPI
RPM	5400
Derived Parameters	
#tracks (N)	1045800
Average bottom track size	2MB
Average top track size	1.6MB



(a) Top-Buffer with Max Size = 3 Tracks (b) Top-Buffer with Two Tracks Given up (c) Top-Buffer + Block-Swap

Figure 3: Illustration of Top-Buffer and Block-Swap.

original locations. The track to be cleaned is selected in a round-robin fashion. Compared to other types of data reclamation policies like Least Frequently Used (LFU) or Least Recently Used (LRU), SCP is able to reclaim a continuous space so that the Top-Buffer region will never be fragmented. Thus, further write requests redirected to the Top-Buffer will not be fragmented either. Also, SCP has less metadata overhead since we neither maintain a candidate list to determine the victim blocks nor update the metadata after a write/read hit. However, some frequently updated data blocks may be cleaned early with SCP thus slightly harming performance.

Top-Buffer introduces a certain amount of overhead in two ways. First, we build a block-level mapping to index buffered blocks. Every read/write to non-free bottom tracks will be converted to reads/writes to both the mapping table and the data. Second, since the Top-Buffer occupies unallocated top tracks, the bottom tracks overlapped by these previously unallocated top tracks will now have an increased rewrite penalty as more valid data blocks have to be protected.

We reduce the overhead of Top-Buffer with the following designs. First, we limit the size of the Top-Buffer to be at most $B_{max}\%$ of the TG, even if there are more unallocated top tracks available. This buffer size limit can reduce the maximum size of the mapping table and allow it to be kept in memory (*memory table budget*) for fast searching. Therefore, the mapping table I/O for the data already buffered is eliminated. For a write to a bottom track block that is not yet buffered, the mapping entry needs to be made persistent before the data is redirected to the buffer. We put the mapping table entries for each Top-Buffer track at the end of the track (1.6KB/track). By locating the buffered data and the corresponding mapping entries in the same track, the seek time for making persistent mapping entries is reduced. Second, as we limit the size of the Top-Buffer, only a small portion of the bottom tracks will have an increased rewrite penalty.

In our current implementation, we set $B_{max}\% = 2\%$, which only needs a memory table budget of 0.004% of the TG size (assuming 4KB sectors) but can still capture a good amount of trace locality. The value of $B_{max}\%$ in a real environment depends on the system's available memory space. Top-Buffer currently only uses unallocated top tracks from the TG end, and we leave potentially better buffer track selection polices as future work.

When used space increases to where there are no more unallocated tracks outside the Top-Buffer, Top-Buffer will clean using SCP and give up tracks for user data (Fig. 3b). As TG usage further increases, the Top-Buffer size keeps decreasing. An extremely small Top-Buffer will have very limited benefits or even worse performance than the Three-Phase baseline.

2.3 Block Swapping (Block-Swap)

To improve performance when the size of the Top-Buffer decreases, we propose block swapping (Block-Swap) which progressively swaps hot (frequently updated) bottom track blocks with cold (infrequently updated) top track blocks. With Block-Swap, fewer updates will go to the buffer and cleaning cost is reduced.

When used simultaneously, Block-Swap occurs during SCP eviction of the Top-Buffer. Hot bottom blocks chosen from SCP victim blocks are swapped with cold top blocks selected from the TG.

Compared to Top-Buffer, Block-Swap has higher I/O overhead, so the hot bottom and cold top blocks to be swapped should be carefully selected. Hot data blocks are selected from the victim blocks evicted from Top-Buffer during SCP. A victim block is considered hot and will be selected to be swapped with a top track block if both of the following conditions are satisfied: 1) the access count is greater than a threshold C; and 2) it belongs to the top T hottest blocks in the victim block list. If either of the conditions fails, the victim block will be considered cold and be written back to its original location. To amortize the extra overhead of swapping, which is eight I/Os (one read and one write each for the top block, the bottom block, and its two neighboring top tracks), we set the access count threshold C to eight. To further minimize the swapping overhead, T is set to 10%. C and T could adapt automatically to the workload, but we leave the investigation of an adaptive algorithm as future work.

To determine hot blocks that reside in the mapping table, we maintain an update count for each mapping table entry and increment a count number when the corresponding block is updated. While a hot bottom block is easy to keep track of, finding a cold block to swap is not a trivial task. We cannot maintain a full table of access counts for all blocks in the TG since the table will be too large. For cold blocks, we design a heuristic random selection algorithm which distributes the tracks into sev-

eral buckets with disjoint ranges of update counts. Continuously cold blocks will be selected for swapping from a randomly chosen track in the bucket with the lowest range of update counts.

During the swapping, a hot bottom block and cold top block pair will be read and then written to each other's location using in-place update. Two new mapping entries will be created to record the physical locations of the two blocks. It is unrealistic to construct an all-to-all block level mapping for the TG. We make a design decision that the total size of the Block-Swap mapping table combined with the Top-Buffer mapping table will be bounded by a memory table budget size to be cached in the memory (see Sec. 2.2). When the unallocated space is greater than $B_{max}\%$ of the TG, Top-Buffer takes all of the mapping table budget and Block-Swap is not used. Block-Swap will start when TG usage goes beyond $1-B_{max}\%$, which is when Top-Buffer starts to give up tracks to user data and consumes less mapping table size from the total budget. Fig. 3c shows an example where a bottom track has four blocks and a top track has three, and Top-Buffer and Block-Swap share the memory budget of the mapping table size. To keep the mapping simple, we only select blocks that have not been swapped, i.e., a block that has already been swapped once will not be chosen to swap with another block. If a previously swapped pair of blocks both become hot, they will first be unswapped and then the hot bottom one will be swapped with a different cold top block.

The Block-Swap mapping entry is updated and synchronized to the disk after the swapping of the two data blocks is completed. In our design, the persistent locations for the Block-Swap mapping entries are distributed into the TG at the end of top tracks. If the Top-Buffer size limit is set to be 2% of the TG, at most one top block is needed for every 30 tracks (15 top and 15 bottom). Once the block that stores the mapping entries is full, the top data blocks in those 15 top tracks will not be selected as cold blocks for swapping.

A trade-off in our current design is that writes that trigger SCP have higher I/O latency, especially when the TG is near full and the expensive Block-Swap occurs during SCP. How to bound this tail latency is left as future work.

2.4 Data Management Design in IMR

In DM-IMR, the tracks are addressed and allocated in the order specified by the Three-Phase baseline (Sec. 1). For a usage U where $0 \leq U \leq 56\%$, writes are directly issued to the disk as there is not yet valid data on the top tracks. When $56\% < U \leq 1-B_{max}\%$, the last $B_{max}\%$ of the TG is used as the Top-Buffer to accommodate updates to the non-free bottom tracks. If $1-B_{max} < U < 100\%$, all remaining unallocated tracks are used as the Top-Buffer, and Block-Swap starts swapping hot bottom data with cold top data. In this range of usage, the unallocated top track capacity shrinks as the data usage grows, and the Top-Buffer has to give the buffer space back to the user data through cleaning. Meanwhile, more pairs of blocks can be swapped because Top-Buffer takes up less of the memory mapping table budget and Block-Swap can consume more mapping table entries. Eventually, when $U = 100\%$, Top-Buffer is off and only Block-Swapping is used. Data will be directly written using in-place update. Here, as there is no space for Top-Buffer, Block-Swap will maintain a virtual Top-Buffer that only collects update statistics to aid swapping decisions.

3 Evaluation

3.1 Implementation

Since no real IMR products are available, we built an IMR simulator where top and bottom tracks have different areal densities and data rates as described in [8] (see Table 1). The I/O latencies are calculated with equations extracted from DiskSim [15].

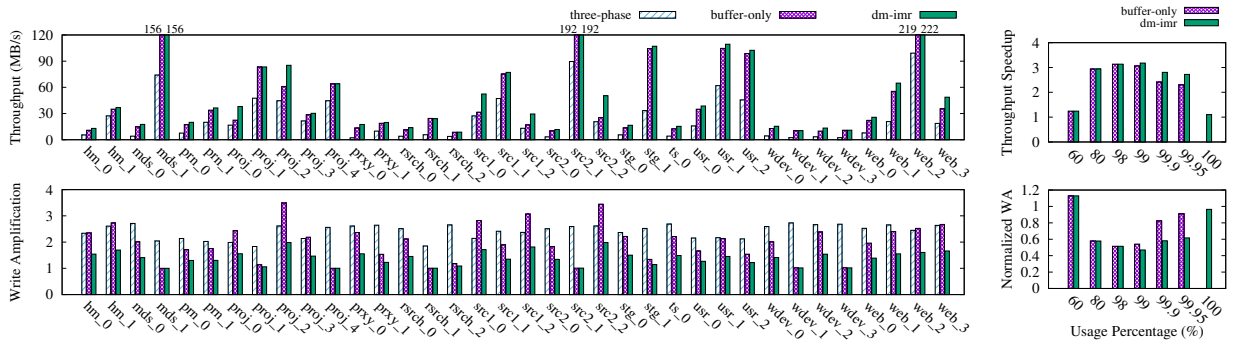
We implement DM-IMR and two baseline approaches: Three-Phase and Buffer-Only. The Buffer-Only scheme uses Top-Buffer without Block-Swap when $56\% < U < 100\%$. Buffer-Only also has a buffer size limit of $B_{max}\%$ ($B_{max}\% = 2\%$ in our evaluation). Block size is set to 512 bytes to align with the Microsoft Research (MSR) Cambridge traces [16] used for evaluation.

3.2 Results

We measure two metrics, average throughput and write amplification (WA), when comparing DM-IMR with the baseline schemes in different TG usages. Here WA is the amount of data actually written to the disk divided by the amount written by the user.

All the MSR traces are tested comparing Three-Phase, Buffer-Only, and DM-IMR (Fig. 4). Fig. 4b plots the average speedup and the normalized average WA of DM-IMR and Buffer-Only relative to the Three-Phase baseline (as 1 in the plots) for all traces with different TG usages ranging from 60% to 100%. Usages under 56% are not used as the performance is identical. Fig. 4a shows detailed throughput and WA plots for each trace at 99.9% TG usage to illustrate the unnormalized performance differences between Buffer-Only, DM-IMR, and Three-Phase for various workloads.

In Fig. 4a, we can see all traces have increased throughputs from Three-Phase to Buffer-Only and further to DM-IMR. The best speedup of DM-IMR from Three-Phase is 7.78× (`prxy_0`), and the WA reduction can be 62.8% (`wdev_1`) for some write intensive traces. There are different throughput speedups for two reasons: 1) different traces have different write/read ratios, and read intensive workloads (e.g., `hm_1`, write ratio 4.7%, speedup 1.35×) will not have as much speedup as write intensive ones (e.g., `prxy_0`, write ratio 96.9%); and 2) the writes have different levels of locality, and traces with



(a) Throughput and WA for usage = 99.9%

(b) Avg Results

Figure 4: Test Results for Throughput and Write Amplification.

low write locality (e.g., `src1_2`, hit ratio 5.1%, speedup 2.24×) will benefit less from DM-IMR than those with good locality (e.g., `prxy_0`, hit ratio 40.4%). We can observe in the bottom graph of Fig. 4a that DM-IMR reduces the WA in all traces. Some traces have higher WAs in Buffer-Only yet have greater throughputs than Three-Phase (e.g., `proj_0`). In this case, although Buffer-Only rewrites more data than Three-Phase, it does so in fewer, larger, consecutive writes. The benefit of fewer random writes negates the downside of more data being written and even brings performance gain.

In Fig. 4b, we can see that DM-IMR has performance benefit across a wide range of TG usages. DM-IMR and Buffer-Only have identical performance when unallocated top tracks are greater than $B_{max}\% = 2\%$ of the TG since they both solely use Top-Buffer. Their performances start to diverge when the usage goes beyond 98%. The WA of Buffer-Only starts to grow due to a decreasing Top-Buffer capacity, and thus the performance begins to drop. In contrast, DM-IMR starts using Block-Swap and can still keep up the performance gain while maintaining low WA when TG is nearly full. Thanks to Block-Swap, at 100% usage DM-IMR can still improve the performance (10.1% higher throughput, 3.5% less WA than the Three-Phase baseline), while Buffer-Only is not able to operate.

The performance gain of DM-IMR over Three-Phase is not significant at low usage (e.g., 60%). When the usage is low, there are many unallocated top tracks and the rewrite penalty of updating a bottom track is small or even zero for free bottom tracks. In this case, there is not much margin for improvement. Additionally, Top-Buffer causes extra I/Os (e.g., mapping table flushes and data migration) and an increased rewrite penalty for bottom tracks that are covered by the Top-Buffer region.

4 Related Work

There have been several existing works on data handling algorithms for SMR [17–30]. However, for the emerging IMR, we only find works by Gao et al. [7, 14], where two data management schemes are proposed: a two-phase and a three-phase implementation. In their two-phase implementation, bottom tracks are first allocated and then the top tracks. In their three-phase implementation, while bottom tracks are allocated during the first phase in the same way as the two-phase implementation, only alternating top tracks are allocated in the second phase. There is a final third phase allocating space from the remaining top tracks. In both their two-phase and three-phase implementation, tracks are allocated in the same radial direction. Our work first proposes a Three-Phase baseline that differs from their three-phase implementation by reversing the second phase allocation direction to preserve data locality. Further, whereas their three-phase implementation maps data blocks statically, our DM-IMR can adapt to the access pattern of dynamic workloads by buffering updates to the unallocated top tracks (Top-Buffer) and swapping hot bottom track data with the cold top track data (Block-Swap).

5 Conclusion

IMR is a recently proposed technology that can have a higher areal data density with much less rewrite penalty compared with SMR when used with HAMR. In this paper, we investigate how IMR can be used with even less write amplification and better throughput. We first propose a Three-Phase baseline and then present our scheme, DM-IMR, which enhances the Three-Phase baseline with two key techniques: Top-Buffer and Block-Swap. Evaluations with Microsoft Research Cambridge traces show that DM-IMR can increase the throughput and reduce the write amplification for all traces when compared with the Three-Phase baseline approach.

Acknowledgments

This work is partially supported by NSF I/UCRC Center for Research in Intelligent Storage (CRIS) and the US National Science Foundation under awards: 1525617, 1439622, 1305237, and 1421913.

References

[1] Eric Brewer, Lawrence Ying, Lawrence Greenfield, Robert Cypher, and Theodore Tso. Disks for data centers. *FAST16, Google, Feb*, 2016.

[2] Robert E Rottmayer, Sharat Batra, Dorothea Buechel, William Challener, Julius Hohlfeld, Yukiko Kubota, Lei Li, Bin Lu, Christophe Mihalcea, Keith Mountfield, et al. Heat-assisted magnetic recording. *Magnetics, IEEE Transactions on*, 42(10):2417–2421, 2006.

[3] Mark H Kryder, Edward C Gage, Terry W McDaniel, William Challener, Robert E Rottmayer, Ganping Ju, Yiao-Tee Hsia, M Fatih Erden, et al. Heat assisted magnetic recording. *Proceedings of the IEEE*, 96(11):1810–1835, 2008.

[4] Jian-Gang Zhu, Xiaochun Zhu, and Yuhui Tang. Microwave assisted magnetic recording. *IEEE Transactions on Magnetics*, 44(1):125–131, 2008.

[5] Jian-Gang Zhu and Yiming Wang. Microwave assisted magnetic recording utilizing perpendicular spin torque oscillator with switchable perpendicular electrodes. *IEEE Transactions on Magnetics*, 46(3):751–757, 2010.

[6] Euiseok Hwang, Jongseung Park, Richard Rauschmayer, and Bruce Wilson. Interlaced magnetic recording. *IEEE Transactions on Magnetics*, 53(4):1–7, 2017.

[7] Kaizhong Gao, Wenzhong Zhu, and Edward Gage. Interlaced magnetic recording, August 8 2017. US Patent 9,728,206.

[8] Steven Granz, Wenzhong Zhu, Edmun Chian Song Seng, Utt Heng Kan, Chris Rea, Ganping Ju, Jan-Ulrich Thiele, Tim Rausch, and Edward C Gage. Heat-assisted interlaced magnetic recording. *IEEE Transactions on Magnetics*, 2017.

[9] Alexander Krichevsky. Heat assisted magnetic recording with-interlaced high-power heated and low-power heated tracks, August 4 2015. US Patent 9,099,103.

[10] Roger Wood, Mason Williams, Aleksandar Kavcic, and Jim Miles. The feasibility of magnetic recording at 10 terabits per square inch on conventional media. *IEEE Transactions on Magnetics.*, 45(2):917–923, 2009.

[11] Ikuya Tagawa and Mason Williams. High density data-storage using shinglewrite. In *Proceedings of the IEEE International Magnetics Conference*, 2009.

[12] Garth Gibson and Milo 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.

[13] Kaizhong Gao. Drive architecture for hard disk drive. *IEEE Magnetics Letters*, 2018.

[14] Kaizhong Gao, Wenzhong Zhu, and Edward Gage. Write management for interlaced magnetic recording devices, November 29 2016. US Patent 9,508,362.

[15] John S Bucy, Jiri Schindler, Steven W Schlosser, and Gregory R Ganger. The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). *Parallel Data Laboratory*, page 26, 2008.

[16] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)*, 4(3):10, 2008.

[17] Ahmed Amer, Darrell DE Long, Ethan L Miller, J-F Paris, and SJT Schwarz. Design issues for a shingled write disk system. In *Proc. MSST'10, Incline Village, NV, USA.*, 2010.

[18] Ahmed Amer, JoAnne Holliday, Darrell DE Long, Ethan L Miller, Jehan-François Pâris, and Thomas Schwarz. Data management and layout for shingled magnetic recording. *IEEE Transactions on Magnetics*, 47(10):3691–3697, 2011.

[19] Yuval Cassuto, Marco AA Sanvido, Cyril Guyot, David R Hall, and Zvonimir Z Bandic. Indirection systems for shingled-recording disk drives. In *MSST'10, Incline Village, NV, USA.*, 2010.

[20] David Hall, John H Marcos, and Jonathan D Coker. Data handling algorithms for autonomous shingled magnetic recording hdds. *IEEE Transactions on Magnetics*, 48(5):1777–1781, 2012.

[21] Chung-I Lin, Dongchul Park, Weiping He, and David HC 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.

[22] Dongchul Park, Chung-I Lin, and David HC Du. H-swd: A novel shingled write disk scheme based on hot and cold data identification. In *FAST12, San Jose, CA, USA.*, 2012.

[23] Weiping He and David HC Du. Novel address mappings for shingled write disks. In *Proc. HotStorage'14,Philadelphia, PA, USA.*, 2014.

[24] Weiping He and David HC Du. Smart: An approach to shingled magnetic recording translation. In *15th USENIX Conference on File and Storage Technologies (FAST17)*, 2017.

[25] Fenggang Wu, Ming-Chang Yang, Ziqi Fan, Baoquan Zhang, Xiongzi Ge, and David H.C. Du. Evaluating host aware smr drives. In *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[26] Fenggang Wu, Ziqi Fan, Ming-Chang Yang, Baoquan Zhang, Xiongzi Ge, and David HC Du. Performance evaluation of host aware shingled magnetic recording (ha-smr) drives. *IEEE Transactions on Computers*, 66(11):1932–1945, 2017.

[27] Saurabh Kadekodi, Swapnil Pimpale, and Garth A Gibson. Caveat-scriptor: write anywhere shingled disks. In *Proc. HotStorage'15, Santa Clara, CA, USA.*, 2015.

[28] Adam Manzanares, Noah Watkins, Cyril Guyot, Damien LeMoal, Carlos Maltzahn, and Zvonimr Bandic. Zea, a data management approach for smr. In *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[29] Peter Macko, Xiongzi Ge, J Kelley, D Slik, et al. Smore: A cold data object store for smr drives. In *Proceedings of the 33rd International Conference on Massive Storage Systems and Technology (MSST'17)*, 2017.

[30] Peter Macko, Xiongzi Ge, John Haskins Jr, James Kelley, David Slik, Keith A Smith, and Maxim G Smith. Smore: A cold data object store for smr drives (extended version). *arXiv preprint arXiv:1705.09701*, 2017.