Caveat-Scriptor: Write Anywhere Shingled Disks

Saurabh Kadekodi, Swapnil Pimpale, Garth A. Gibson
Carnegie Mellon University

1 Introduction

The increasing ubiquity of NAND flash storage is forcing magnetic disks to accelerate the rate at which they lower price per stored bit. Magnetic recording technologists have begun to pack tracks so closely that writing one track cannot avoid disturbing the information stored in adjacent tracks [13]. Specifically, the downstream track will be at least partially overwritten, or *shingled* by each write, as shown in Figure 1, and the upstream track will tolerate only a limited number of adjacent writes. Some data that was stored in the downstream track will be lost, forcing firmware or software to ensure that there was no data in those locations that might be read in the future.

In order to avoid deployment obstacles inherent in asking host software to change before *shingled disks* can be used, the current generation of shingled disks follow the model established by flash storage: a *shingled translation layer* of firmware in the disk remaps data writes to empty tracks and cleans (read, move, write) fragmented regions to create empty tracks. Known as *Drive-Managed Shingled Disks* [9], host software does not need to change because disk firmware will do extra work to cope with any write pattern that could destroy data. To reduce or eliminate this extra work, API changes in the hard disk have been proposed [1] to enable Host-Managed management of shingled disks.

This paper explores two models for *Host-Managed Shingled Disk* operation. The first, Strict-Append, breaks the disk into fixed sized bands and compels disk writes to occur strictly sequentially in each band, allowing only per-band-truncate-to-empty commands to recover space. This is approximately a physical realization of the classic Log-Structured File System (LFS) [21], and shares the need for the file system to schedule and execute cleaning of bands. The second model, Caveat-Scriptor (latin for "let the writer beware"), exposes a traditional disk address space and a few shingled disks parameters: a distance in the downstream block address space that is guaranteed to never experience shingled overwrite data loss and a distance in the upstream block address space that cannot tolerate multiple adjacent writes. Host-Managed software for Caveat-Scriptor shingled disks is allowed to write anywhere, but if it fails to respect these distance parameters, it may destroy data. We show in this paper that Caveat-Scriptor enables the reuse of previously written and deleted data to perform *free cleaning* (explained in Section 4.2), deferring background cleaning by hours and also requiring far less total cleaning as compared to Strict-Append. Because sophisticated log-structured cleaning algorithms have been extensively studied, we do not anticipate a huge difference in throughput. Instead we expect a reduced probability of very long response times to be the primary benefit of less cleaning.

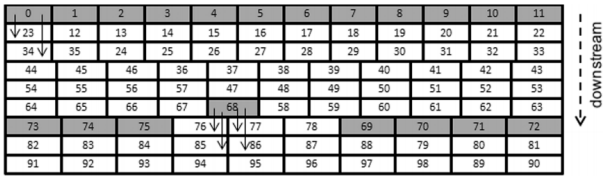

Figure 1: Feldman's [9] Figure 5 showing the effect of two writes, one to logical block address (LBA) 0 and one to LBA 68, on an example shingled disk. Arrows point to downstream LBAs that will be damaged and shading shows logically downstream addresses that will not be damaged, for LBAs 0 and 68.

In this paper we will present a simple model for Host-Managed Caveat-Scriptor, describe a simple FUSE-based file system for Host-Managed Caveat-Scriptor, construct and describe a file system aging tool and report initial performance comparisons between Strict-Append and Caveat-Scriptor. We will show the potential for Caveat-Scriptor to help limit heavy tail response times for shingled disks.

2 Related Work

The viability of a shingled translation layer, akin to a NAND flash translation layer, was noted when Shingled Magnetic Recording (SMR) was proposed for commercialization [11]. Most shingled disk layout research to date pursued Strict-Append variants of LFS [21]. Amer et al. [6, 5] mapped bands to LFS segments and treated each band as a circular log. Cassuto et al. [7] proposed two indirections: (1) random-write zones on disk which act as a writeback cache leading later to a read-modify-write band rewrite cycle. A more sophisticated technique (2) involved creating logically contiguous circular write-back buffers on disk, called S-Blocks. Hall et al. [14] built on top of S-Blocks to aid random writes. They defined large sequential shingled runs holding most of the

data, I-Regions, each with a small circular buffer called an E-Region. E-regions acted as writeback caches for I-Regions and enabled background cleaning of I-Regions. Lin et al. [17] exploited the difference between hot and cold data on disks. They segregated hot from cold to prevent costly cleaning operations on hot data which would naturally be deleted more frequently. They also explored temperature aware garbage collection mechanisms.

At least two file systems have been built for banded shingled disks. SFS [17] was designed for video servers, and assumed 64 MB bands, and the presence of both random and sequential shingled zones on the same disk. HiSMRfs [15] supported append-only semantics, relied on unshingled partitions and added a RAID module to support striping across multiple shingled drives.

Recently, Aghayev and Desnoyers [3] reverse engineered a Drive-Managed shingled disk via a series of carefully crafted microbenchmarks and video recordings of the resulting disk arm movement. The disk behavior indicated the presence of an on-disk cache accepting incoming requests that were lazily written back to their final destination. Also, the shingled disk they analyzed seemed to have bands ranging from 15-40 MB across the disk's various zones.

3 Shingled Disk Model

3.1 Strict-Append

Strict-Append is a type of Host-Managed SMR [9] that restricts host writes to only occur at the write cursors of bands. The write cursor points to the next sector that can be written to in a band. Writes implicitly move the write cursor forward, that is, the only write operation is append. When a band is cleaned, due to the inter-track interference (see DPID in Section 3.2), the write cursor is reset to the first sector of the band; it cannot be moved back to any other location except the start of the band. Bands are separated from one another by a band gap.

3.2 Caveat-Scriptor

Caveat-Scriptor is also a type of Host-Managed SMR [9, 22] in which no write address restrictions are enforced. Instead the host is aware of drive characteristics that enable it to make safe data placement decisions. Caveat-Scriptor summarizes all layout risks in a few per-drive factory-set parameters:

- **Drive Isolation Distance (DID):** When writing to a certain LBA (k), shingling may result in damage to other LBAs. Call the distance to the largest damaged LBA associated with the chosen LBA (k) the *isolation distance* of k. DID refers to the largest isolation distance observed for any LBA on the disk. It is safe to assume that each write damages no more than DID LBAs downstream. [1]
- **Drive Prefix Isolation Distance (DPID):** Absent from Feldman's model for Caveat-Scriptor [9] was a simple parameter to protect upstream inter-track interference degradation of stored data. To remedy this absence, we define DPID as the largest number of preceeding LBAs that may be degraded by a specific LBA write, and require that no more than one write should be done within DPID LBAs downstream of data that may be read in the future.

For example, in 2015, DPID is likely to be about 1 track (on the order of a megabyte) and DID is likely to be on the order of 1-3 tracks (a few megabytes). These numbers will probably increase slowly in the future.

4 SMRfs

SMRfs is a simple FUSE-based file system designed for experimenting with shingled disk Host-Managed APIs. It is not intended for production use. It implements both Strict-Append and Caveat-Scriptor and runs on top of a (traditional) raw disk partition. SMRfs assumes each Host-Managed shingled disk offers two partitions, one unshingled (traditional) and one shingled. The unshingled partition allows random access and is used to store SMRfs metadata, while the 100-1000X bigger shingled partition stores data.

SMRfs formats the small unshingled partition as a traditional file system whose files have no data in the unshingled partition. Each file is split into blocks of 1 MB. Data blocks are allocated in the shingled partition using a simple next-fit policy. The location of each block is recorded as an extended attribute associated with the stub file on the unshingled partition. We use Ext4 for the unshingled partition.

SMRfs has limited internal parallelism; there is a background thread for reading and writing blocks on the shingled partition, a background thread for cleaning fragmented space on the shingled partition and a foreground thread for executing FUSE-relayed commands. SMRfs also has an internal cache with limited functionality (e.g. currently an open file must be completely instantiated in the cache).

4.1 SMRfs for Strict-Append

SMRfs on a Strict-Append Host-Managed shingled disk implements a variant of LFS [21], made simpler because all metadata is in the unshingled partition. LFS regions are the size of a band, 32 MB in our experiments [3]. Our cleaning mechanism follows a cost-benefit policy consistent with LFS and also restricts the amount of live data moved in each cleaning cycle to 128 MB, following the advice in Matthews et al. [18]. The cleaner is invoked in the background when the disk is idle and less than 35% of its space is free, and it runs continuously when the disk has less than 5% free space. It is also invoked

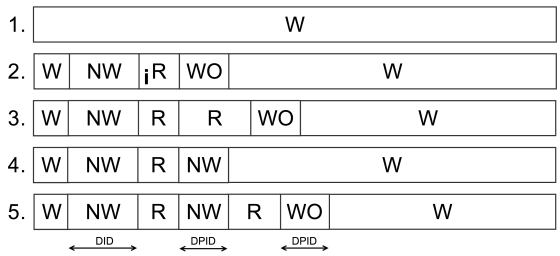

Figure 2: Caveat-Scriptor Operation Example (DID > DPID): (1) The entire disk is writable (W). (2) Allocate and write an extent starting at LBA *i*. DID LBAs before *i* are then not-writable (NW), and DPID LBAs following the extent are write-once (WO). (3) Allocate and write more data sequentially enlarging the readable (R) extent, and shifting the WO safety gap. (4) Delete data just written, freeing up LBAs occupied previously, but also converting the trailing safety gap to NW because the first written extent has been exposed to the allowed single write inter-track interference event, and must be rewritten before an adjacent downstream write is allowed again. (5) Finally, allocate and write as close as possible downstream, fragmenting the disk surface.

synchronously (in the foreground) if there is less unfragmented free space than is needed to accomodate a data block being written.

4.2 SMRfs for Caveat-Scriptor

SMRfs on a Caveat-Scriptor Host-Managed shingled disk implements a traditional disk layout allocator with the following necessary Caveat-Scriptor safety policy: *protect potentially readable data.* Any run of shingled LBAs containing data that may be read again, Readable (R), must be preceeded by at least DID LBAs that are Not-Writable (NW), and followed by either a run of DPID LBAs that are Write-Once (WO), if the most recent write event to any LBA in the WO region was before the oldest write event in the last DPID LBAs in the R run, or followed by a run of DPID LBAs that are NW (if, for example, the tail section of a sequentially written run was deleted from SMRfs, so the tail of the surviving R run has already seen one nearby downstream write as illustrated in Figure 2).

The consequence of this safety policy is that deletion of N LBAs from a larger run of readable LBAs does not allow all N LBAs to become writable. Typically there will be DPID LBAs at the start and DID LBAs at the end that are NW. This reduces writable space, and this reduction becomes worse as the disk is fragmented into smaller R and W runs.

When a deletion causes all data in a readable run of LBAs to never be readable again, SMRfs for Caveat-Scriptor does not need to reserve safety gaps at the beginning and end of the newly deleted run. In fact, it can

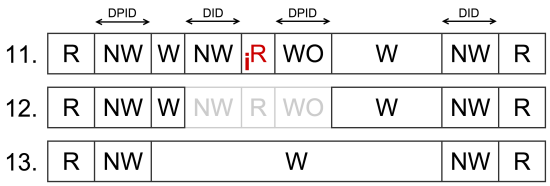

Figure 3: Free Cleaning in Caveat-Scriptor Example (DID = DPID): (11) Starting with an aged and fragmented disk, with 3 runs of R data and 2 runs of W or WO LBAs, delete the entire R run at LBA *i*. (12) Since there is no nearby readable data to protect, the adjacent NW and WO runs can be reclaimed. (13) The entire enclosing writable run is coalesced.

reclaim adjacent NW and WO runs. As illustrated in Figure 3, reclaiming NW or WO safety gaps is *free cleaning*. Strict-Append SMRfs can only do this when all LBAs before the write cursor in a band have been deleted, so free cleaning in Caveat-Scriptor SMRfs is much more likely to happen.

5 Preliminary Evaluation

Our experiments use the PRObE Marmot cluster [10] whose nodes have dual core 64 bit, 1.6 GHz AMD Opteron-242 processors along with 16 GB of RAM and a single 2 TB WDC SATA 7200rpm disk.

Because the behavior of shingled disks is driven by allocation and fragmentation, we built and used an aging tool before running benchmarks [23]. Our aging tool is parameterized to fill a target fraction of the disk with user data (70% of a 100 GB disk) based on a distribution of file sizes (Figure 4 (1)) taken from a study of a Yahoo! Hadoop Cluster [8] and a distribution of file ages (Figure 4 (2)) accelerated (by a factor of 1 sec = 6.08 hrs of the reference distribution) to match an observed age distribution [4]. Long runs of creates and deletes (with probability 0.9) are performed rather than simple alternation while maintaining the target utilization.

Figure 4 (3) shows a 5+ hour test run of Strict-Append SMRfs versus Caveat-Scriptor SMRfs on a 100 GB partition. In both runs, the aging tool follows the same pseudo-random sequence, and in both we inject four application benchmark runs into the middle, at the same point in the aging tool progress (marked at time A). The four benchmarks are two invocations of Postmark [16] and two invocations of Filebench [19].

In the first few minutes, shown in Figure 4 (3), the aging tool achieved 70% utilization of the disk capacity. Strict-Append SMRfs (blue line) always at the top continued to fill the disk, cleaning for the first time when only 5% of the disk was available for new writes. Because the aging tool was running constantly throughout the test, there was never any idle time cleaning, so Strict-Append SMRfs triggered 128 MB of cleaning each

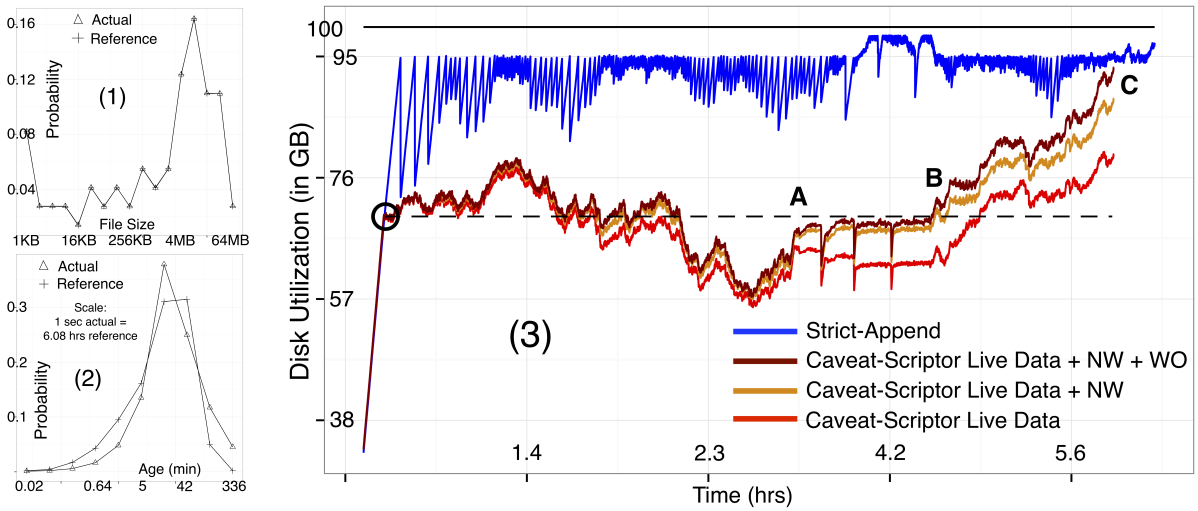

Figure 4: Space availability under test (3) and aging tool file size and file age distributions (1) and (2) (obtained from [4] and [8] respectively). We aged a 100 GB disk to 70% utilization (shown by the dashed line) via successive create + write and delete operations (without fsync) imitating 100% cache hits for reads. At time A, 800 GB worth of aging had been done before starting two successive runs of Postmark [16] (file sizes 3-7 MB) then two successive runs of the Filebench [19] fileserver profile (mean file size 9 MB) while the aging tool was slowed down to half its normal speed. Time B shows the end of the second Filebench run which left 4.5 GB additional user data on the disk. The aging tool then resumed full speed until time C, when Caveat-Scriptor finally ran out of SMRfs block-sized free space. The end point of Strict-Append running the same workload is when it had transfered the same amount of user data as Caveat-Scriptor.

time free disk space fell to 5%. Because 128 MB does not take too long to clean, it only suffered synchronous foreground cleaning during the very busy benchmarking runs.

Caveat-Scriptor SMRfs (red, yellow and brown lines) achieved free cleaning throughout the test; that is, no cleaning work is invoked until time C. The lowest (red) line shows the total amount of user data being mutated by the aging tool; the middle (yellow) line shows the user data plus NW space and the top (brown) line shows the user data plus NW and WO space. Because the aging tool emulates a Hadoop style workload, with large file sizes, free cleaning works quite well over hours. During the benchmarks, however, even with large average file sizes, considerable NW space accumulated (and persisted because the aging tool will never delete benchmark files).

Figure 5 shows the average throughput of the two runs of Postmark (on the right) for Strict-Append versus Caveat-Scriptor. While there was a small average throughput advantage for Caveat-Scriptor, the use of small (128 MB) amounts of cleaning work each time cleaning is triggered in Strict-Append limited the overall slowdown it experienced. On the left in Figure 5, we show the maximum response times of events in the two runs of Filebench. Here the penalty for frequent cleaning in Strict-Append is more evident.

One interpretation for the behavior in Figure 4 (3) is that Caveat-Scriptor, although able to defer the continuous cleaning seen in Strict-Append, is not able to avoid building up space lost to safety gaps beyond 5.5 hours in this test. Eventually disk space will be fragmented into enough small safety gaps that continuous cleaning could be needed in both systems. Like solid state disks (SSD), if the data mutation workload does not leave enough idle system opportunities for extensive cleaning (defragmentation), peak write throughput and response times will degrade.

The difference, however, is that after sufficient disk idleness for defragmentation, should an intense mutation period start again, Caveat-Scriptor SMRfs will be able to operate without cleaning for much longer than Strict-Append. Provided the duty cycle for user work in the shingled disk is not 100% (and disk idleness is usually very common [12]), Caveat-Scriptor may be able to hide all cleaning, while Strict-Append will need a very small duty cycle to hide much cleaning.

6 Discussion and Conclusion

While Strict-Append SMRfs has a sophisticated cleaner based on 20+ years of LFS research, cleaning in Caveat-Scriptor is a new topic for storage research. In addition to free cleaning and the resulting deferred cleaning, Caveat-Scriptor cleaning can be “a little at a time”, potentially causing much less impact on user response times.

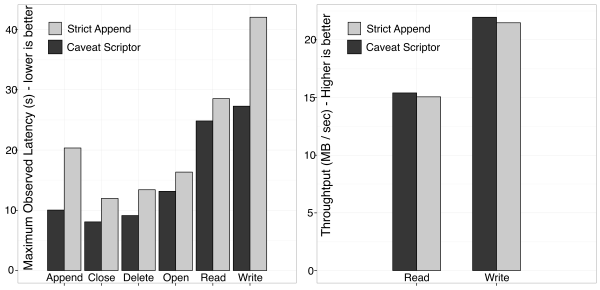

Figure 5: Maximum latency by operation type observed by Filebench (left) when executing in parallel with the aging tool as shown in Figure 4. Average read and write throughput when running Postmark (right) along with the aging tool as shown in Figure 4.

Caveat-Scriptor can provide cleaning flexibility, but it can also enable catastrophic off-by-one software layout bugs. For this reason we expect drive vendors to be slow to offer Caveat-Scriptor APIs for Host-Managed disks. However, even a drive-managed shingled disk needs a disk model for its firmware engineers to program. Perhaps using an internal Caveat-Scriptor API might allow Drive-Managed shingled disks to offer less variable response times.

Finally, Caveat-Scriptor [9] is similar to copy-on-write (CoW) file systems like Btrfs [20] in terms of treatment of in-place updates. With efforts already being made to make Ext4 SMR-friendly [2], incorporating Caveat-Scriptor in a CoW file system seems a feasible next step.

7 Acknowledgements

We would particularly like to thank Timothy Feldman, David Anderson, Michael Miller from Seagate Technology and our shepherd Bernard Wong for their valuable guidance. This research was supported in part by the DOE and Los Alamos National Laboratory, under contract number DE-AC52-06NA25396 subcontract 153593-1 (IRHPIT2), the National Science Foundation under awards CNS-1042537 and CNS-1042543 (PRObE, www.nmc-probe.org), and Seagate through the CMU Data Storage Systems Center.

References

[1] T10 Zoned Block Commands (ZBC). *URL: http://www.t10.org/cgi-bin/ac.pl?t=f&f=zbc-r02.pdf* (2013).

[2] SMR friendly Ext4. *URL: https://github.com/Seagate/SMR_FS-EXT4* (2015).

[3] AGHAYEV, A., AND DESNOYERS, P. Skylight - a window on shingled disk operation. In *FAST 2015* (2015), USENIX.

[4] AGRAWAL, N., BOLOSKY, W. J., DOUCEUR, J. R., AND LORCH, J. R. A five-year study of file-system metadata. *ACM Trans. on Storage (TOS)* (2007).

[5] AMER, A., HOLLIDAY, J., LONG, D. D., MILLER, E. L., PARIS, J., AND SCHWARZ, T. Data management and layout for shingled magnetic recording. *Magnetics, IEEE Trans. on* (2011).

[6] AMER, A., LONG, D. D., MILLER, E. L., PARIS, J.-F., AND SCHWARZ, S. T. Design issues for a shingled write disk system. In *MSST IEEE* (2010).

[7] CASSUTO, Y., SANVIDO, M. A., GUYOT, C., HALL, D. R., AND BANDIC, Z. Z. Indirection systems for shingled-recording disk drives. In *MSST IEEE* (2010).

[8] FAN, B., TANTISIRIROJ, W., XIAO, L., AND GIBSON, G. Diskreduce: Replication as a prelude to erasure coding in data-intensive scalable computing. In *SC11* (2011).

[9] FELDMAN, T., AND GIBSON, G. Shingled magnetic recording areal density increase requires new data management. *USENIX;login* (2013).

[10] GIBSON, G., GRIDER, G., JACOBSON, A., AND LLOYD, W. Probe: A thousand-node experimental cluster for computer systems research. *USENIX; login* (2013).

[11] GIBSON, G., AND POLTE, M. Directions for shingled-write and twodimensional magnetic recording system architectures: Synergies with solid-state disks. *CMU-PDL-09-014* (2009).

[12] GOLDING, R., BOSCH, P., WILKES, J., ASSOCIATION, U., ET AL. Idleness is not sloth. In *USENIX* (1995).

[13] GREAVES, S., KANAI, Y., AND MURAOKA, H. Shingled recording for 2-3 tbit/in2. *IEEE Trans. on Magnetics* (2009).

[14] HALL, D., MARCOS, J. H., AND COKER, J. D. Data handling algorithms for autonomous shingled magnetic recording hdds. *Magnetics, IEEE Trans. on* (2012).

[15] JIN, C., XI, W.-Y., CHING, Z.-Y., HUO, F., AND LIM, C.-T. Hismrfs: A high performance file system for shingled storage array. In *MSST* (2014).

[16] KATCHER, J. Postmark: A new file system benchmark. Tech. rep., Technical Report TR3022, NetApp, 1997.

[17] LIN, C.-I., PARK, D., HE, W., AND DU, D. H. H-swd: Incorporating hot data identification into shingled write disks. In *MASCOTS* (2012).

[18] MATTHEWS, J. N., ROSELLI, D., COSTELLO, A. M., WANG, R. Y., AND ANDERSON, T. E. Improving the performance of log-structured file systems with adaptive methods. In *ACM SOSP* (1997).

[19] MCDOUGALL, R., AND MAURO, J. Filebench. *URL: http://www.nfsv4bat.org/Documents/nasconf/2004/filebench.pdf* (2005).

[20] RODEH, O., BACIK, J., AND MASON, C. Btrfs: The linux b-tree filesystem. *ACM Trans. on Storage (TOS)* (2013).

[21] ROSENBLUM, M., AND OUSTERHOUT, J. K. The design and implementation of a log-structured file system. *ACM Trans. on Computer Systems (TOCS)* (1992).

[22] SINGH, M. K. Comparing performance of different cleaning algorithms for smr disks. Master's thesis, Carnegie Mellon University, 2014.

[23] SMITH, K. A., AND SELTZER, M. I. File system agingincreasing the relevance of file system benchmarks. In *ACM SIGMETRICS* (1997).

Notes

[1] There are some LBAs immediately following a written LBA that are always safely unaffected. The minimum number of these is called Drive No Overlap Range (DNOR) and can be used to dynamically "construct" unshingled space [9]. We do not use DNOR space in Caveat-Scriptor SMRfs yet.