A review of techniques to reduce write/update latency in SMR drives

Shrey Gupta[1], Avani Wildani[1]

[1]*Department of Computer Science, Emory University*

shrey.gupta@emory.edu

avani@mathcs.emory.edu

Abstract— **SMR drives or Shingled Magnetic Recordings are one of the newest addition to the drive technology. It uses shingling of tracks to achieve high areal density. However, the shingling reduces SMR's efficiency as it increases the write/update latency. Although SMR drives are laudable for their original space-efficient architecture, their impediment arises in their throughput for writing or updating the data. In this paper, we review the improvements in the write/update technology for these drives. We will look at how they address this problem and the software methodologies used to overcome it.**

Keywords— **SMR, Shingling, areal, latency**

I. Introduction

Shingled Magnetic Recording (SMR) is a data storage technology that is a recent innovation in the domain of storage recordings/drives [1]. Unlike Conventional Magnetic Recording (CMR) [2] where data is written on tracks that are parallel to each other, an SMR has overlapping tracks that are organized as shingles on a roof. The utility of such an architecture is that it increases the areal density of the disk and ergo the capacity. As observed above, overlapping of the tracks improve the areal density of the disk but also reduces its write and update efficiency. The write head of a disk is usually larger than the read head and hence during the write process, the contents of the bottom track is overwritten as well. This creates a pitfall for random write operation and hence, the SMR drives usually follow the sequential write model.

There have been a few improvements in the domain of reducing latency of write/update in SMR over the years since it was introduced. In Sections IV, V and VI, we would be looking at three methodologies [3,4,5] that addresses this issue.

II. SMR Drive Architecture

As mentioned in the previous section, the tracks of the SMR drive is layered as shingles on a roof. Each track is composed of multiple sectors. The data is written into these sectors. The tracks are sometimes bundled into bands and each such band is separated by a track. On the other hand, the Conventional Magnetic Recordings (CMR) has tracks placed parallel to each other. The write head is usually wider than the read head for CMR's. An SMR architecture is usually implemented on traditional CMR drives. Hence, when a random write or update request is made in an SMR, the write head which is wider than the trimmed track will need to correct the data till the end of the drive or band depending upon the SMR architecture.

III. Drive Parameters

There are three types of mechanisms in a SMR namely, Read, Write and Garbage Collection (GC). GC is a memory management process that reclaims the memory no longer in use. The GC process is of two types: Foreground GC and Background GC. Inside the SMR, there are two queues: external and internal queue. The requests (read/write) arrive at the external queue and await to be passed to the internal queue. If it's a read request, it gets directly passed to the internal buffer. However, the write requests are buffered in the external queue until a certain threshold and sent as one write request at a time to the internal queue.

Type: Rotation Study Report
Final Edit: 12/3/2019

The SMR drive parameters are provided in Table 1. The parameters are predefined for each SMR and elucidate how efficiently the drive performs. The seek time (t_{seek}) is the time taken for the disk head to fly from one cylinder (C1) where it is located to the target cylinder (C2). The total no. of tracks on the drive is given by T_{total}. The average no. of sectors in a track is represented by n_{spt}, and s_{sec} gives the size of the sector in bytes. As mentioned previously, a segment/band is multiple tracks grouped together. The size of a segment is given by s_{seg}. The speed (rotations/min) of the drive and the time taken to complete one rotation/revolution is given by s_{rpm} and t_f respectively.

TABLE I
SMR DRIVE PARAMETERS

Name	Description
$t_{seek, min}$	Minimum seek time
$t_{seek, max}$	Maximum seek time
T_{total}	Total no. of tracks
n_c, n_s	Cylinder no., Surface no.
n_{spt}	Average sector per track.
s_{sec}	Sector size
s_{seg}	Total track no. per segment
s_{rpm}	Rotation per minute
t_f	Full revolution time
K	internal queue size

IV. Most recent update

Since update-induced latency i.e. the latency which occurs when data is updated, depends on the location of the sectors being updated on the drive, the technique provided in [3] tries to reduce the latency overhead by working on data placement. The data that is to be written/updated is placed in such a way that the most recently updated information is stored in the outermost tracks. This allows an easy modification without the cumbersome process of erasing the information from other shingled tracks first and then writing it back to them. The technique used by the paper is known as MRU (Most Recent Update) which is also used for cache data management ([4] - [7]).

The MRU technique stores the most recently updated information in the outermost track on the assumption that the next update request would be of this information. Although, it's a crude methodology to approach the update issue but it is comparatively effective. To illustrate this well, we assume that there are k tracks in a band/segment of SMR. Suppose the update operation requires the 4th track to be updated. Hence, the data from k-4 tracks would be written to a buffer and then the update takes place only for the data to be written back to the drive. Another issue that arises during the update is the timing of the track being written to the outermost position. The most efficient way to go about this is to perform this during the update operation itself. Hence, in the above scenario, the k-4 tracks will be written first and the 4th track will be written to the outermost track of the band. This eliminates any extra overhead.

V. SMRDB

This paper [4] created a database for SMR to overcome the sequential write restrictions. The database is called SMRDB. SMRDB is a Key-Value (KV) database engine for SMR disks. It uses LSM-trees to tackle the write restrictions. SMRDB is similar to LevelDB which is an open source LSM tree based, embeddable, Key-Value database library. The LevelDB and hence, the SMRDB consists of 3 kinds of tables:

1. A log table
2. A memtable
3. A SSTable (sorted string table)

Every KV pair is first written to the *log table* or the *log file*. They are then added to the *memtable*. When the memtable gets full, it's contents are added to the SSTable. The size of a memtable is equivalent to the size of shingled band. Basically, each band in an SMR are made up of multiple SSTables, and when a memtable gets full, it's contents are transferred to an empty band.

The SSTables on the other hand are organized into levels. Each level has a threshold for the size and every level is bigger than the previous one. Also note that each level has a non-overlapping key range i.e. keys in one level are unique to that level and do not occur in other levels. However, the L0 level has overlapping keys with the other levels since the KV pairs in this level are a result of memset table dumps or dumps from the memset table. Hence, if we want to search for a KV pair, we would have to read the entire L0 level and each non-zero level's first key. Therefore, we can see that the number of searches also decrease using this level-wise approach.

The question still persists as to how the SMRDB reduces the write induced latency. This is achieved using both the memset table and the SSTable. The memset table stores the write data and gets emptied when it is full. This in turn decreases the latency of the incoming write requests. The SSTable on the other hand helps to easily search the right level for the write or write-update because of it's non-overlapping keys format.

Conclusion

There is a growing need for quintessential storage devices- low cost, large capacity and high throughput. Shingled Magnetic Recordings are a step closer to this utopia for storage disks. But like every other storage device they have their limitations. However, their novel architecture provides a hope for breaking ground for newer technologies in this domain. The Most Recent Update technique [3] provides us a direction of how cache management techniques can be used for mitigating the latency. We would be exploring more of such techniques to determine if they push the boundaries for reducing the write/update latency.

As [4] mentions, the SMR Database i.e. SMRDB does not require a file system. This eliminates the need for drive managed SMR architecture or presence of a file system that recognises SMR drives. The SMRDB can be deployed directly on host-managed SMR disks as well as traditional disks.

References

[1] T. Feldman and G. Gibson, "Shingled Magnetic Recording," vol. 38, no. 3, p. 9, 2013.

[2] S. Iwasaki and Y. Nakamura, "An analysis for the magnetization mode for high density magnetic recording," *IEEE Transactions on Magnetics*, vol. 13, no. 5, pp. 1272–1277, Sep. 1977.

[3] Venkataraman, Kalyana Sundaram, et al. "Techniques Mitigating Update-Induced Latency Overhead in Shingled Magnetic Recording." *IEEE Transactions on Magnetics*, vol. 48, no. 5, 2012, pp. 1899–1905., doi:10.1109/tmag.2011.2178099.

[4] Pitchumani, Rekha, et al. "Smrdb." *Proceedings of the 8th ACM International Systems and Storage Conference on - SYSTOR '15*, 2015, doi:10.1145/2757667.2757680.

[5] Chen, Shuo-Han, et al. "A New Sequential-Write-Constrained Cache Management to Mitigate Write Amplification for SMR Drives." *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing - SAC '19*, 2019, doi:10.1145/3297280.3297336.

[6] He, Weiping, and David H C Du. "SMaRT: An Approach to Shingled Magnetic Recording Translation," n.d., 15.

[7] K. So and R. N. Rechtschaffen, "Cache operations by MRU change," IEEE Trans. Comput., vol. 37, no. 6, pp. 700–709, Jun. 1988.

[8] B. Ozden, R. Rastogi, and A. Silberschatz, "Buffer replacement algorithms for multimedia storage systems," in Proc. IEEE Multimedia, 1996, pp. 172–180.

[9] M. Kandemir, F. Li, M. J. Irwin, and S. W. Son, "A novel migration based NUCA design for chip multiprocessors," in Proc. IEEE/ACM Conf. High Performance Computing, Networking, Storage and Analysis, Austin, TX, Aug. 2008.

[10] J. Jeong and M. Dubois, "Cache replacement algorithms with nonuniform miss costs," IEEE Trans. Comput., vol. 55, no. 4, pp. 353–365, Apr. 2006.