This article appeared in the March/April 2015 issue of *FreeBSD Journal*®

Shingled [SMR] Magnetic Recording

[This Is Not Your Father's Disk Drive]

by Justin Gibbs

Existing hard disk products are nearing the estimated 1Tb/in^2 limit in areal density achievable using current perpendicular magnetic recording (PMR) technology. While alternatives to PMR are being developed, manufacturers are adopting a more compact track layout, termed shingled magnetic recording (SMR), to increase capacity. SMR currently provides a 20% improvement in density, but removes the ability to independently update sectors on the media. In this article we explore the characteristics of SMR and its impact on traditional storage software.

Since the introduction of the hard drive by IBM in 1956, the disk industry has been tracking Moore's Law, doubling storage density every two years. But that trend may now be in jeopardy. Disk platters are nearing the 1Tb/in^2 areal density limit of the currently used technology, perpendicular magnetic recording (PMR), and higher density schemes such as heat-assisted magnetic recording (HAMR) and bit patterned media (BPM) are still two years or more away.

To fill this roadmap gap, manufacturers are turning to shingled magnetic recording (SMR). SMR achieves a higher density by shrinking the effective track size. Doing this without any changes to the media or head design takes advantage of the existing PMR technology's ability to read smaller tracks than it can write. Instead of leaving a gap between two adjoining tracks to ensure there is no interference (Figures 1 and 2), SMR purposely overwrites a portion of the previously written, neighboring track when laying down a new stripe of data.

The old track remains, but is trimmed to a narrower width. The new track is the same width as on a non-SMR drive. Writing in this way across a platter, with an occasional gap band so writes don't have to be continuous across the entire media, currently yields a 20% increase in capacity. (Figure 3) This advantage may increase in the future.

SMR is so called because writes are performed in a pattern similar to the shingles on a roof. Just as it is impossible on a roof to replace a single shingle without perturbing those surrounding it, SMR drives do not allow operations that replace the contents of individual sectors. Write operations must start at a gap band on the media and proceed sequentially to the next gap band in order to not lose previously written data.

Since this model is so different from the behavior of disk drives recorded with independent tracks, drive manufacturers are adding firmware features to ease the transition for file systems and applications. In products shipping today, the drive manages all the complexity of shingling and emulates a standard direct access device. Models available within the next year will provide additional commands to expose the SMR traits of the drive. SMR-aware software can use this information to avoid operations that must be emulated, but the drive will still accept any direct access command. For environments where the whole software stack is SMR aware, models will eventually be offered with no emulation at all. With the extra CPU, memory, and media over-provisioning required for emulation removed, these drive models should provide the best price per terabyte.

The emulation provided by current SMR models allows all existing software to function, but often at a significant performance penalty. Writing randomly to a shingled device will force the firmware to perform garbage collec-

Fig. 1 Conventional Track/Sector Layout

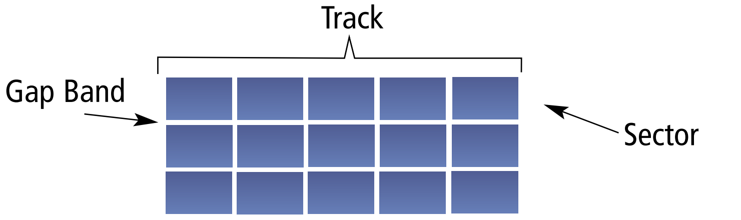


Fig. 2 Conventional Media Layout

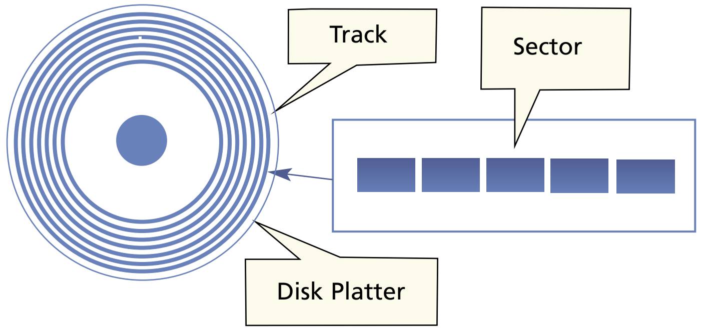
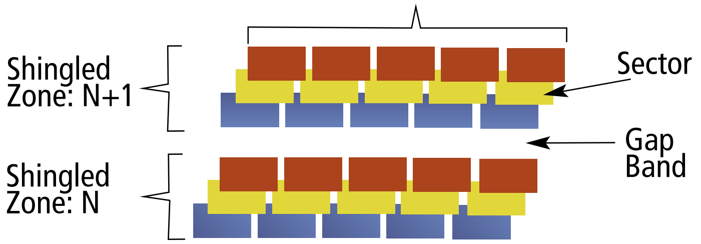


Fig. 3 Shingled Zone/Track/Sector Layout



tion, and in most cases, a single write operation from the host will be converted into two or more write operations to the media. Given enough media over-provisioning, RAM, or flash, it should be possible to provide performance that meets or exceeds that of a traditionally formatted hard drive. But along with increasing density, the market expects decreasing per-terabyte cost.

The additional component cost would break this trend. So while emulation will improve over time, the storage systems looking to achieve optimal performance will need to become SMR aware.

Adapting existing storage stacks to SMR will not be easy—the ability to overwrite any LBA is an inherent assumption of most implementations. Given this difficulty, many have hoped that SMR will be phased out once new technology like HAMR and BPR are available. Currently, the chance that this will occur seems remote. The roadmaps of the major players in the spinning drive market show SMR as a contributing technology far into the future for their march to achieve Moore's Law.

SMR Devices

Current industry roadmaps show three distinct types of SMR devices: drive managed, host aware, and host managed. Drive-managed devices provide the full direct access command set of a standard drive and use internal mapping techniques to give the illusion that individual sectors can be updated randomly. Host-aware devices include all the emulation support of a drive-managed device, but also support commands that SMR-aware software can use to write optimally and avoid emulation penalties. Host-managed devices remove all emulation support, forcing the host to manage tasks like garbage collection directly. In all cases, the physical characteristics of an SMR drive are the same.

An SMR hard drive is divided up into zones, collections of sequentially addressed sectors. Zones can be managed independently since they are separated from each other by unwritten space. These "gap bands" ensure that write activity from one zone cannot impact another. Zones can be either conventionally formatted or shingled. A conventional zone allows updates to individual sectors, just like on a conventional, non-SMR, disk drive. A shingled zone must be written sequentially from lowest to highest logical block address (LBA-sector address) so that already in-place data isn't corrupted by subsequent writes. In order to achieve the density improvements SMR promises, SMR zones are large—tens if not hundreds of megabytes—and the majority of zones in a device must be shingled. Typical configurations may also include a small number of conventional zones for storing rapidly changing blocks and/or metadata for managing the SMR zones, but this is not required (Figure 4).

Drive-Managed Devices

Providing conventional hard drive semantics while using SMR media adds tremendously to the complexity of the device's firmware. One possible strategy for performing this emulation is to completely decouple the LBA presented to the host from the physical location of data on the media. This technique is used to great effect on solid state drives, and SMR zones have many of the same traits as the flash memory used in SSDs: a shingled zone is effectively the erase block size, and writing in ascending address order is highly preferred. However, the economics do not support this approach. The market expects spinning drives to be significantly cheaper than flash storage. Currently an 8-TB SMR drive retails for under $.04 per GB, compared to almost $.40 per GB for the lowest performance 1-TB flash drive. There isn't enough price margin to allow manufacturers to add the large RAM, flash, or CPU resources needed to manage a fully dynamic LBA map on an SMR device. Even if the market supported higher prices, the latency cost for random-access reads on spinning media gives good incentive to make most requests that are sequential to the host (sequential in LBA space) also sequential on the media.

Fig. 4 SMR Media Layout

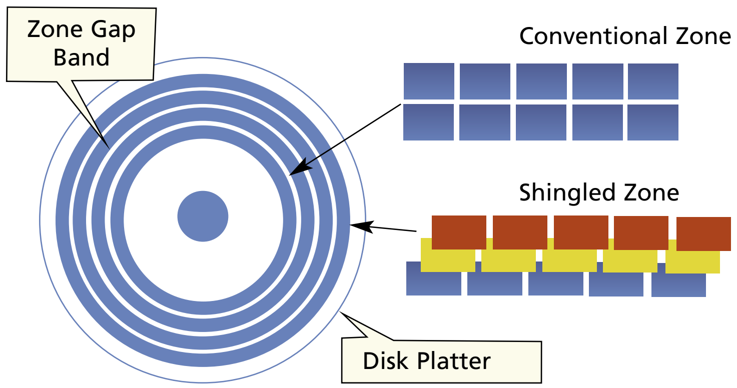
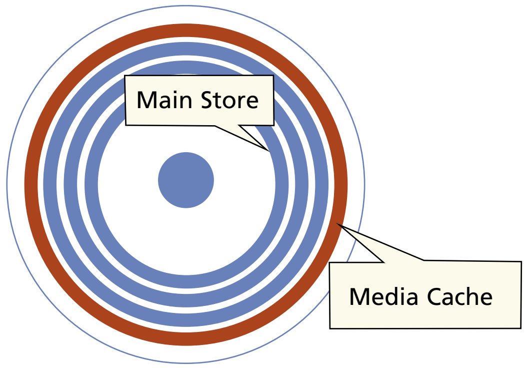


Fig. 5 Layout with "Media Cache"

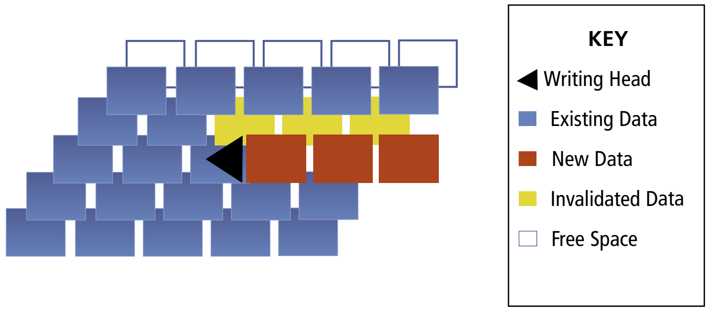


Instead, current generation drives use a "cache and clean" approach. Writes that would violate the sequential write requirements of an SMR zone are directed to a media cache. This media cache is composed of zones reserved for use by the firmware. These zones typically occupy space at the outer diameter of the platters where transfer rates are the greatest (Figure 5). An implementation may also reserve zones throughout the media in order to avoid large seek penalties for diverting to the media cache when the head is already positioned at the inner diameter. The media cache may utilize either conventionally formatted zones, shingled zones, or a combination of both.

Using shingled zones to manage other shingled zones may seem counterproductive, but doing so increases the density of the cache, resulting in less raw capacity being lost to emulation. Drives that chose this approach apply the same log-structured, journaling techniques used in databases, flash devices, and even some conventional file systems to always write sequentially to the cache.

The steps taken to service a read request help to illustrate how the media cache and the remaining SMR zones interact in this emulated environment. The LBAs requested by the read fall into two categories: those residing in the main SMR store and those that were redirected to the media cache and may be migrated back to the main store sometime in the future. To make this classification, the firmware maintains enough information in RAM to quickly determine if an LBA has been redirected to the media cache. If that lookup succeeds, the physical location in the media cache is known. If the lookup fails, the physical location is also known. Bad block remapping aside, there is a linear mapping from LBA to physical address in the main store, allowing the zone and zone offset to be quickly derived from the LBA. Once all physical locations are known, the disk drive can perform seek optimization, read the blocks, and stitch them together for transmission to the host.

Fig. 6 Mid-Zone Overwrite

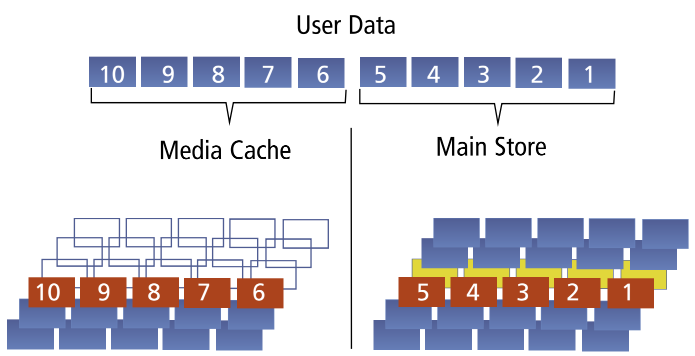


By maintaining a map of only those sectors that reside in the media cache, the RAM and CPU resources required for emulation can be greatly reduced. The media cache is a small fraction of the device's total capacity and so is the map. But this also means that it is possible to exhaust the media cache. This is where the clean step of "cache and clean" comes into play. In parallel with servicing host activity that causes reads and writes to the media cache, the drive must also promote data from the media cache to the main store, freeing space to service future write activity.

In its most basic form, cleaning entails preserving any data to be retained from a main SMR zone in the media cache, overwriting the SMR zone with both new and retained data, and then marking portions of the media cache as free for reuse. There are, however, strategies the firmware can employ to make the process less expensive. To reduce the impact of cleaning operations on host command latency, the cleaning can be done in stages. If host I/O arrives, those LBAs that have been transferred to the main store can be marked as free, and the cleaning operation suspended. After servicing higher-priority operations, cleaning can resume where it left off. Another savings is possible given that writes to an LBA in the middle of a zone can impact LBAs at higher addresses, but not at lower addresses. The firmware can thus avoid preserving data at LBAs below the first LBA of data being promoted from the media cache. Additionally, the range of LBAs disturbed by a mid-zone write is relatively small (a track or two) and well known to the firmware. Rather than clean all the way to the end of the zone, the firmware may stop early and just leave the LBAs for the invalidated sectors in the media cache (Figure 6). You can think of this approach as "hole filling." If the range of contiguous LBAs to be promoted to the media cache is greater than the number of LBAs invalidated by a write, hole filling results in a net reduction in the amount of media cache in use.

There are workloads where having a media cache increases performance. If, for example, the host is rapidly issuing random writes to a small number of LBAs, these sectors can simply reside in the media cache without ever being promoted. Since most media cache implementations are journaled, the LBAs from these writes are written sequentially to the media. The net result is near sequential I/O performance for bursts of random writes. Even when promotion must eventually occur, if the total load on the device is low enough, the drive may have sufficient idle time to do promotion without any host-visible penalty. However, random write workloads are migrating to solid-state devices. For the more common spinning-disk workload of storing larger objects that are rarely overwritten, the goal should be to avoid the media cache entirely and eliminate the multiple-write penalty incurred due to promotion.

Fig. 7 **Hole Filling/Redirection to Media Cache**



To avoid the media cache on a drive-managed device, the host must issue write operations so it is easy for the drive firmware to compute the benefit of writing directly to the main store. The first step is to ensure that operations that are sequential in LBA space are presented to the drive that way. Write commands to adjoining LBA regions should be issued from lowest to highest LBA without any interleaved commands. The native queue depth varies depending on the command protocol being used, but in general, a spinning drive can only process a handful of commands concurrently. It may fail to reorder out-of-order commands and detect that they are sequential. Similarly, any delay between sequential commands may cause the firmware to timeout and assume that the stream has ended.

Issuing commands in the right order is required, but not sufficient to bypass the media cache. The goal of the firmware is to retire a write without having to read any data. It can only do that and also bypass the media cache if the write operation is large. At today's media densities, a track holds on the order of 4MB, and at least a track is invalidated by a mid-zone write. Any write operation (or group of sequential write operations) that consumes less than two tracks of space will go to the media cache. For writes larger than this, the drive will write the first portion directly to the main store and direct the last track's worth of data to the media cache (Figure 7). In this way, the drive can fulfill the write without having to read any data—the write to the main store is stopped before invalidating any of the sectors that need to be retained.

The need to meet data integrity guarantees can also prevent writing directly to the main store. It is possible for the host to disable a drive's write cache, forcing all write commands to be reported as complete only after the data is committed to stable storage (persists across power loss). Given the limited queue depth and RAM resources of the drive, this will almost certainly cause the firmware to prematurely terminate a direct media write so that it can finalize commands and return them to the host. Selectively issuing write commands that bypass the cache (e.g., setting the "force unit access" bit in a SCSI write) or a command to flush the full cache will also force the drive to end any current optimization. For these reasons, the write cache should always be enabled on drive-managed SMR devices and the frequency of synchronizing commands heavily limited.

Host-Aware Devices

Even more strategies for avoiding the media cache become available to the host when dealing with a host-aware device. These drives support two additional commands: REPORT ZONES, and RESET WRITE POINTER. REPORT ZONES does what the name implies. With it, the host can determine the number, type, and size of all host visible zones. REPORT ZONES also provides the current position of the write pointer within shingled zones. The write pointer marks the end of valid data in a zone and is the location that should next be written in order to avoid an emulation penalty. The RESET WRITE POINTER command allows modification of the write pointer in a zone. Currently, this command is specified to only allow the write pointer to be set to the beginning of a zone, effectively erasing a zone in one shot. Based on the reserved byte space in this command, it appears that future versions of the ATA and SCSI specifications may allow the write pointer offset to be explicitly set—for example, to just erase then overwrite the tail end of a zone.

With this geometry information in hand, the host can intelligently assign LBAs to the data it is writing. Data structures for tracking free and allocated space can be aligned and sized to match a drive's native zone layout. Rapidly overwritten data such as a file system's free space maps can be directed to conventional zones. Large writes of user data can be allocated from contiguous space starting at the write pointer in SMR zones. As LBAs are freed from SMR zones, the free space fragmentation this causes can be tracked so that this space is only reused when the system is relatively idle or when there is a high return when compared to the cost of reclaiming a zone. When all the data in an SMR zone is no longer needed, a RESET WRITE POINTER will tell the drive it needn't preserve any sectors for subsequent writes to that zone. These are just a few of the possible optimizations for avoiding nonsequential writes and the penalties for having the drive emulate them.

Host-Managed Devices

With sufficient sophistication in the host software, we can take these optimizations to their logical conclusion: perform all zone management within the host. This is where the host-managed devices in the SMR roadmap come into play. While host-aware devices report shingled zones that are "sequential write preferred," on a host-managed device, these are "sequential-write mandatory." All emulation features are absent on this drive type and any write that doesn't start at the write pointer for a zone is rejected.

There are a few unexpected complications with host-managed devices. While the firmware of a drive-managed or host-aware device is free to invalidate LBAs in the middle of a shingled zone so long as access to those LBAs is properly redirected to the media cache, the current specification for host-managed devices is much more restrictive. To ensure the drive can report which LBAs are valid utilizing the least amount of drive-maintained metadata, writes are only allowed at the write pointer, and only those LBAs below the write pointer are readable. Any attempt to read outside the range from the beginning of the zone to the write pointer will fail, even if those sectors were written previously and are still valid on the media. For this reason, host-aware software is somewhat more constrained in the approaches it can take to effectively utilize a host-managed device.

The specifications for host access to SMR drives are in their infancy, and we can expect some aspects of this restriction to eventually be addressed. Already there is a proposal to allow a circular zone type. In this type of zone, only LBAs in the interference region just past the write pointer are considered invalid. All other sectors can be read. While still not providing the freedom to implement some of the redirection schemes of drive-managed and host-aware devices, circular zones would naturally lend themselves to journaling. And journaling may be sufficient for getting optimum performance from these devices.

Conclusion

After almost 60 years, the fundamental way that software interacts with spinning magnetic media is changing. Given the trajectory of the main providers of hard disks, storage systems developers have little choice but to adapt. Until they do, consumers should expect reduced performance from SMR drives when deployed in traditional workloads.

The stage is now set to see how storage appliances, operating systems, and file systems tackle the challenge of SMR. With an understanding of how drive-managed devices perform their emulation, some basic optimization for SMR drives is possible. Going beyond this and taking full advantage of host-aware and host-managed devices will take more time. Storage stacks are complicated—often taking tens to hundreds of man-years to develop. Because of this, the impact of SMR will proceed in slow motion through the storage industry. Five years from now, there will be definite winners and losers: those who have adapted to SMR and those who have not. For now, it's just too early to predict who the casualties will be. ●

Justin Gibbs is the founder and president of the FreeBSD Foundation, and has been working on the storage-related subsystems of FreeBSD since 1993. He currently works at Spectra Logic Corporation building petabyte-scale, archive storage systems using FreeBSD, flash, disk, and tape.