Next-generation storage interfaces: Zoned Block Devices

CSCI 333
Williams College
This Video: Zoned Storage
(and related topics)

- (Abbreviated recap) Hard Disk Drives
  - Basic Design/Geometry
  - Performance characteristics
- Shingled Magnetic Recording
  - Concepts and interface
  - Position in the storage stack
- Other SMR Interfaces/Opportunities
- IMR
- ZNS NVMe extensions (Zoned SSDs)
Hard Disk Drives (HDDs)

- High capacity, low cost
- Predictable performance
  - **Unwritten contract**: LBAs near each other are more efficient to access than LBAs that are far away
HDDs

- Sector (unit of transfer)
- Disk Head (seeks in/out)
- Tracks (concentric circles)
- Platters (rotate)
Performance Observations

- **Setup** (placing the disk head) is expensive $O(10 \text{ ms})$
  - **seeking** to target track
  - Up to a full **rotational delay** to locate target sector
  - Once the disk head is in place, data **transfer** is quite fast $O(100s \text{ MiB/s})$
**Performance Goal**: build a system where data is written sequentially (i.e., no random writes)
Keeping HDDs Relevant

- HDDs compete on $/GiB, not performance
- As capacity goes up, $/GiB down
- **Problem:**
  - Capacity gains traditionally result of reduced track width to increase density
  - Physical limits restrict our ability to shrink tracks further
- We’re stuck… unless?
Shingled Magnetic Recording (SMR)

- Increases HDD density by overlapping tracks

Perpendicular Magnetic Recording
Shingled Magnetic Recording (SMR)

- Increases HDD density by overlapping tracks

**Insight:** Read head is more precise than write head

**Technique:** Overlap next track, but leave enough of “lower” track visible for safe reading
SMR Introduces Challenges

- **Writing** data becomes harder: append-only
  - No random writes
  - No overwrites
  - Must garbage collect to reclaim space
No Random Writes

If we don’t write to zones *append-only*, we could lose data
No Overwrites

Must perform **out-of-place updates**, or suffer a read-modify-write of entire zone
Garbage Collection

1. Copy **live** data from source to destination
2. Reclaim old zone
Garbage Collection

1. Copy **live** data from source to destination
2. Reclaim old zone
Recall HDD Observations

• **Problem**: Seeking is slow

• **Solution**: perform large sequential I/Os

**Takeaway**: HDD *performance optimizations* translate into SMR *correctness requirements*
Implementing SMR Logic
Question: who enforces the SMR write constraints?
Drive Managed vs. Host Managed

- **Easy to Deploy**
- **Limited HW resources**

- **Flexible**
- **Consumes host resources**
Hardware/Software Interface: Zoned Block Commands

Two types of zones

- **Conventional Zones**
  - Random write capabilities of “normal” disks

- **Sequential-write-required zones**
  - Each zone has a single *write pointer*
    - Append blocks to zone’s write pointer
    - Reset zone write pointer (reclaim space)
Other HDD Opportunities

- Other SMR interfaces have been proposed
  - Caveat Scriptor [Kadekodi ’15]
  - Configurable zone layouts (Flex) [Feldman ’18]
- Interlaced Magnetic Recording (IMR)
  - Combines HAMR and overlapping tracks
Caveat Scriptor

[Kadekodi '15 HotStorage]

Basic Idea:

• Drive characteristics are exposed to the user

• User can write anywhere, but data may be lost if user doesn’t manage data carefully

Caveat Scriptor means “let the writer beware”
Interlaced Magnetic Recording

[Interlaced Magnetic Recording]

- Each top track overlaps two adjacent bottom tracks.
- Writing to a bottom track would corrupt neighboring top tracks.
  - Unlike an SMR zone, this disruption is limited to immediate neighbors, rather than requiring rewriting entire zones.

Figure 3: Depiction of interlaced track recording

[Interlaced Magnetic Recording]
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.

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.

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 \( \ll \) 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 total space is filled. In the third phase, the remaining space will be allocated from the remaining top tracks so that the top and bottom tracks are filled in alternating order.

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

[Wu ’18 HotStorage]
Open Questions

• Translation layer design

• Garbage collection schemes

• *MR-aware applications (SMR/IMR)?
  • Key-value stores
    ‣ Integrating *MR maintenance with DS work
  • File systems
    ‣ Changing disk formats & write patterns
Let’s Think About Designs: Translation Policy

What are our options? I.e., what is the design space?

- **Static or dynamic mappings from LBA->PBA?**
  - What do you think is done in practice?
    - Skylight [Aghayev ’15] designed & performed benchmarks to tease out drive parameters for DM-SMR drives
Let’s Think About Designs: Translation Logic Location

What are our options? I.e., what is the design space?

- Application, file system, or dedicated translation layer?
  - + The more you specialize, the more you can optimize
  - - The more you specialize, the narrow your use case

- Research has produced SMR-specific key-value stores (GearDB, FAST '19), file systems (Evolving ext4 for Shingled Disks, FAST ’17), archival storage arrays (Pelican, Microsoft Research)

- Commodity “archive” products are all secretly DM-SMR
What About SSDs?
Review: SSDs

• Interface:
  • Read pages
    ‣ As many times as we want
  • Program pages (write)
    ‣ Once -> then need to erase before rewriting
    ‣ Limited endurance -> need to wear level
  • Erase whole blocks
    ‣ Erasing is slow
    ‣ Need to perform GC -> migrate live data
• FTL plays a role in all of these tasks: wears many hats
  • L2P page translation, wear leveling, GC, ECC, …
Zoned Namespaces

• If you squint your eyes, the SMR issues look a lot like the constraints that we faced when discussing SSDs

• The SSD approach was for FTLs to manage the write/erase constraints in firmware, similar to DM-SMR

• **Observation**: a large ecosystem of HM-SMR software could “just work” on SSDs if the interfaces were aligned

• But what parts of the FTL should migrate “out” to software?
Zoned Namespaces

- Some things seem hard and very hardware specific
  - ECC is not something I think we can write portably or efficiently without low-level HW knowledge…
- But ZNS spec lets us handle the rest in software
  - Zones are similar to SMR zones
  - In ZNS SSDs, we implement wear leveling, mappings from LBA->PBA, and GC
Zoned Namespaces

- Not yet widely available, but it is possible (in theory) to buy ZNS devices today

- Question: Do you want one of these in your laptop?

- Question: Who stands to benefit the most from ZNS devices?
Takeaways

• As technologies evolve, legacy interfaces restrict our ability to optimize for new features

• But as we add new features, legacy software needs to be rewritten to accommodate

• Translation layers let us bridge the gap, but there is an open question of where to put them?

• Building logic into applications is expensive and not portable, but it maximizes our ability to optimize