Next-generation storage interfaces: Zoned Block Devices

CSCI 333
Williams College

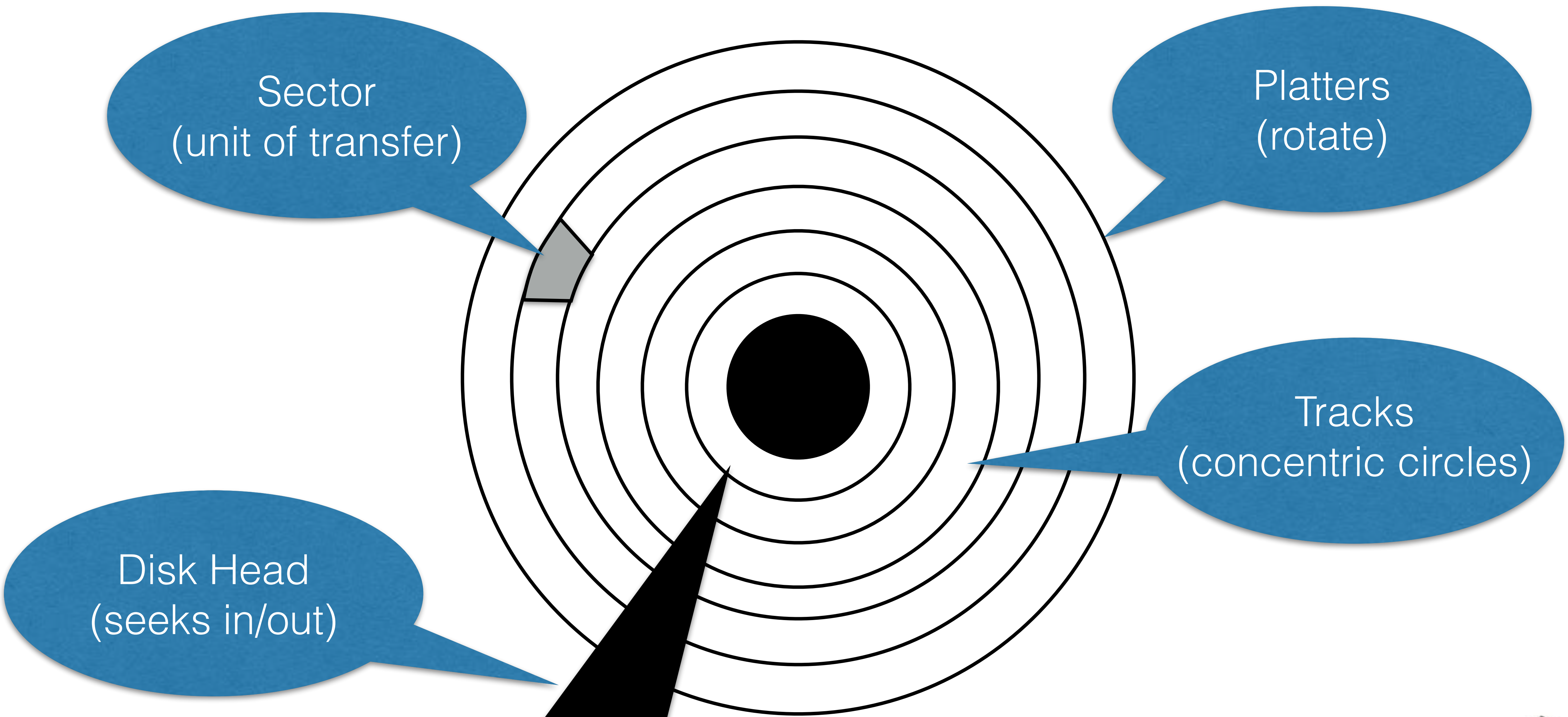
This Video: Zoned Storage
(and related topics)

- (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)

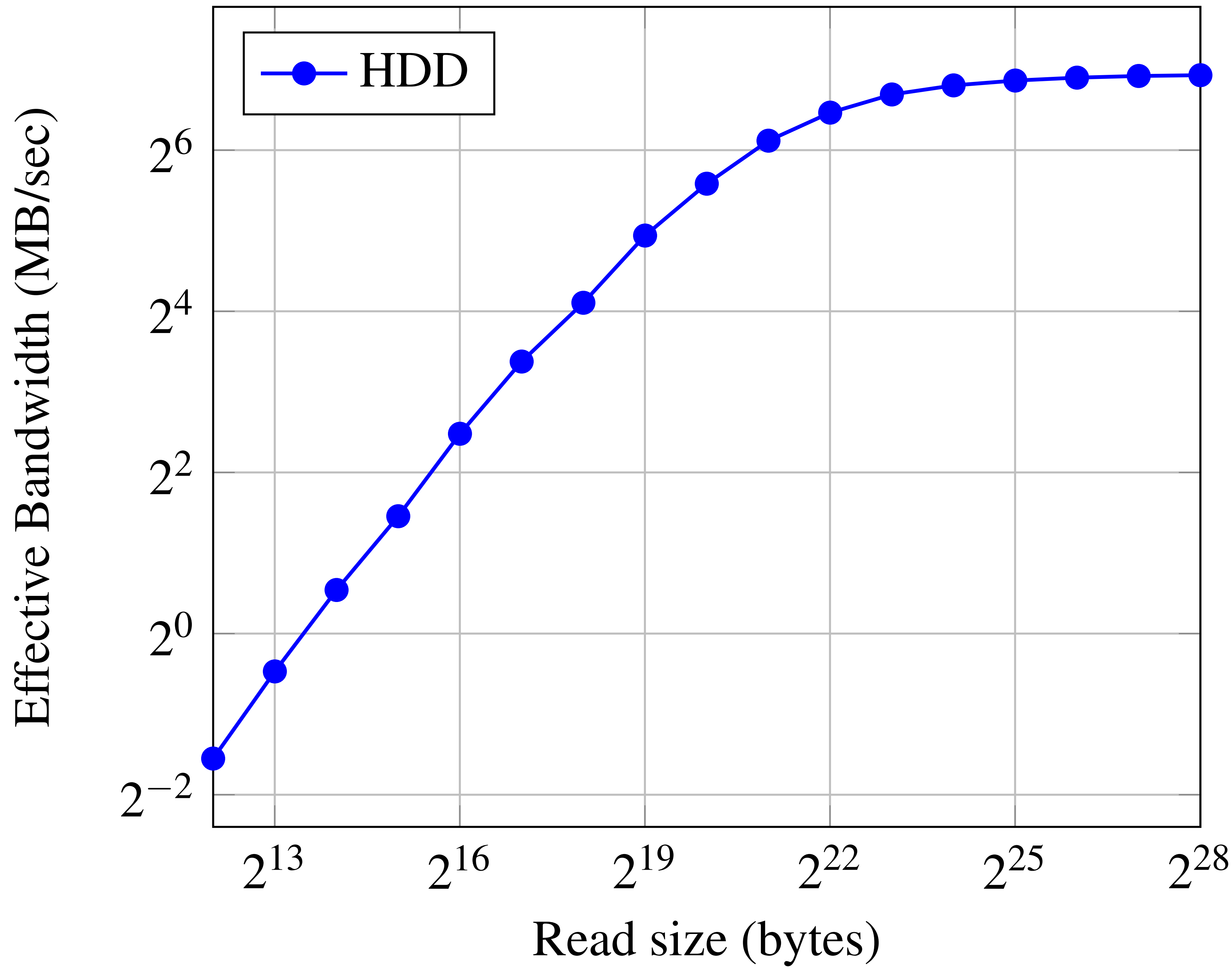
- High capacity, low cost
- Predictable performance
  - Unwritten contract: LBAs near each other are more efficient to access than LBAs that are far away

HDDs


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

Performance Observations

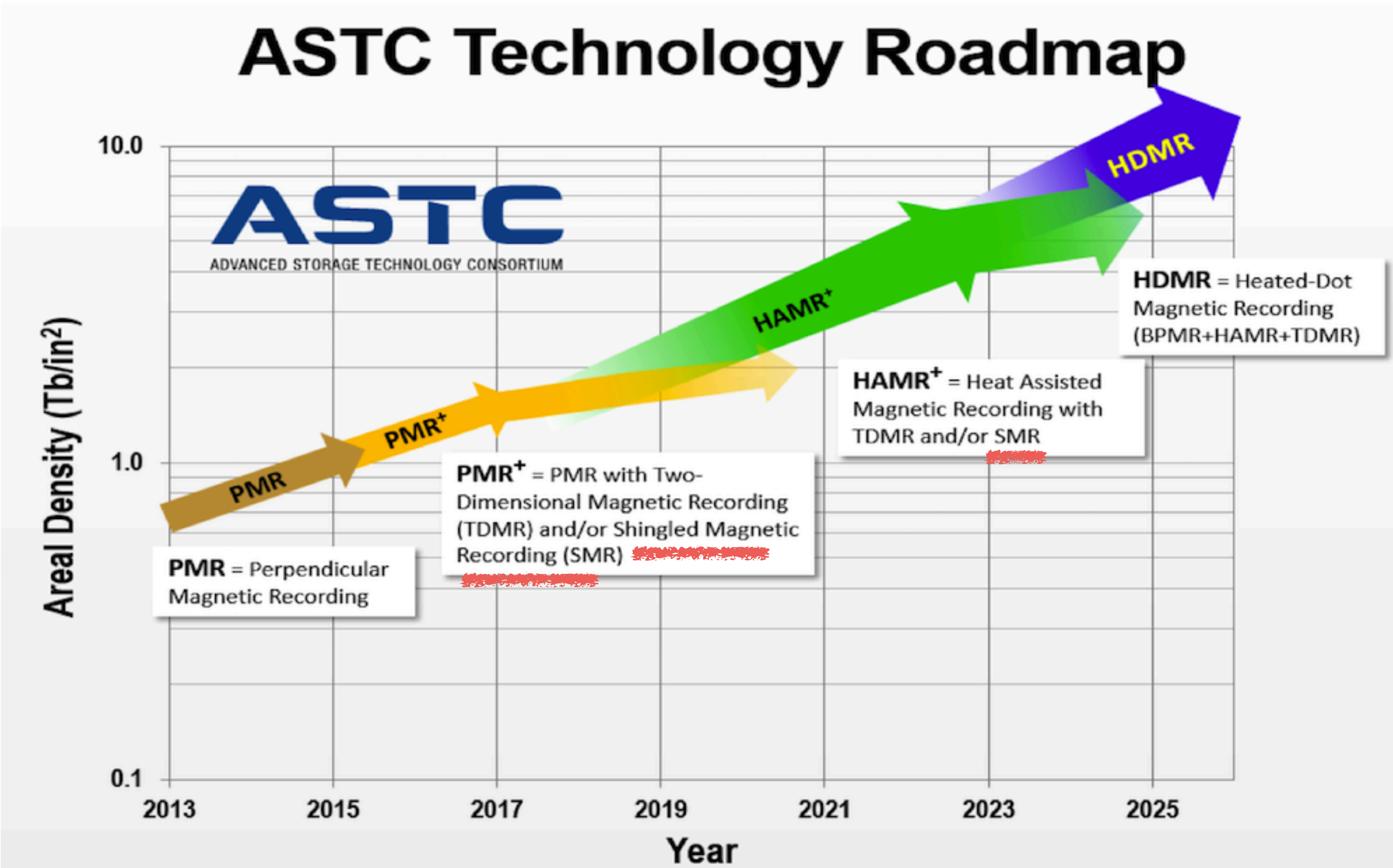
- **Setup** (placing the disk head) is expensive O(10 ms)
  - seeking to target track
  - Up to a full rotational delay to locate target sector
- Once the disk head is in place, data **transfer** is quite fast O(100s MiB/s)



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?

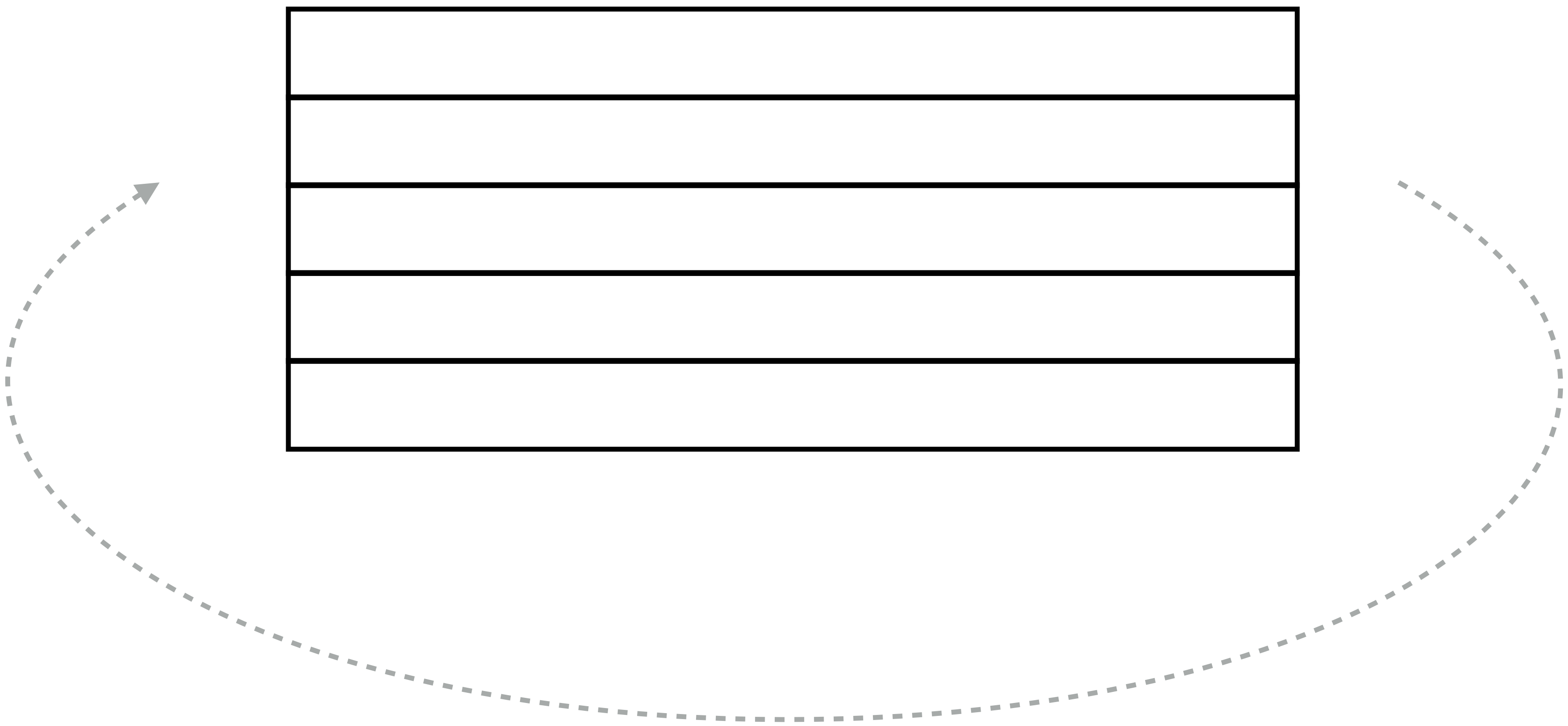


[https://blog.seagate.com/craftsman-ship/hamr-next-leap-forward-now/]

Shingled Magnetic Recording (SMR)

- Increases HDD density by overlapping tracks

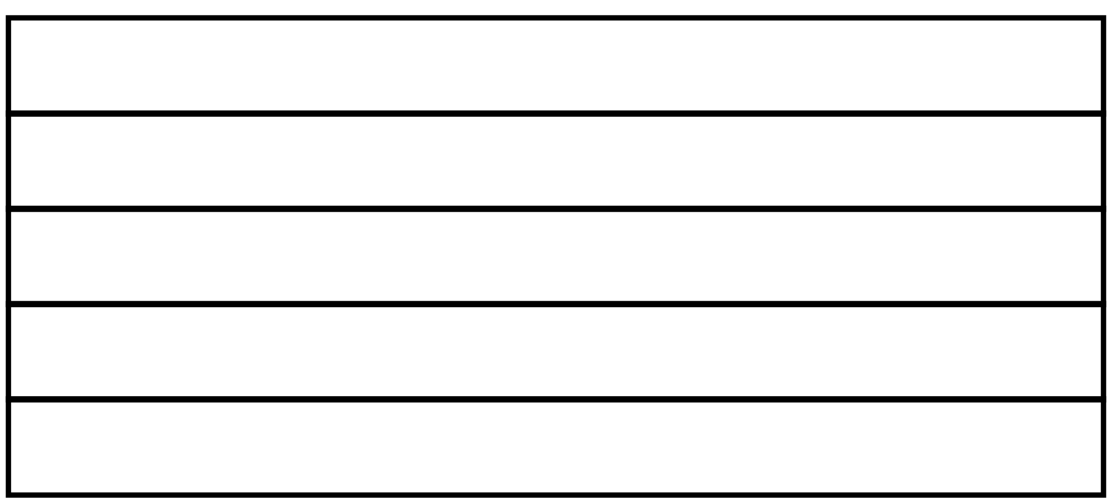
Perpendicular Magnetic
Recording



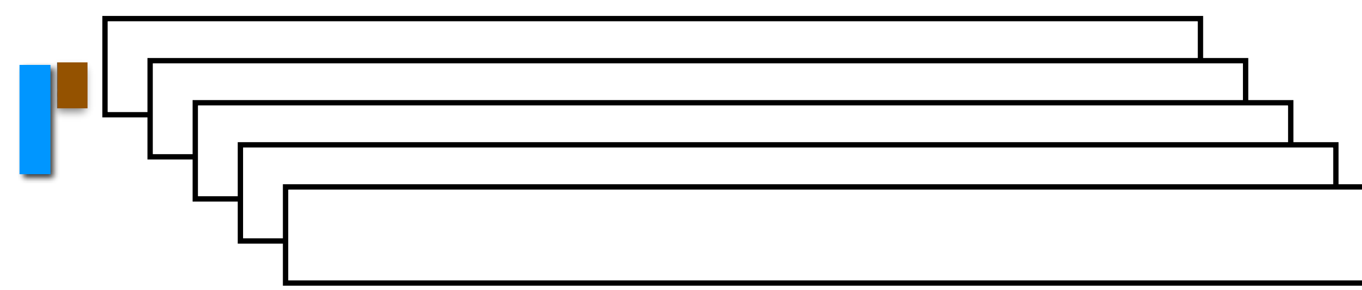
Shingled Magnetic Recording (SMR)

- Increases HDD density by overlapping tracks

Perpendicular Magnetic Recording



Shingled Magnetic Recording

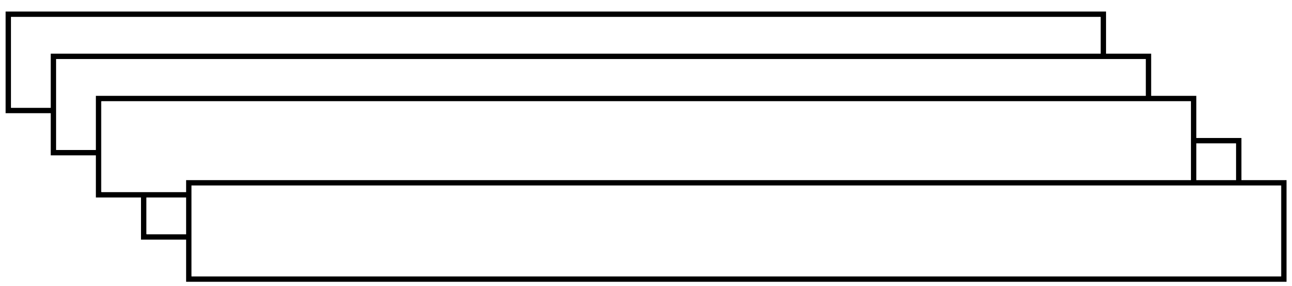


- **Insight:** Read head is more precise than write head
- **Technique:** Overlap next track, but leave enough of “lower” track visible for safe reading

SMR Introduces Challenges

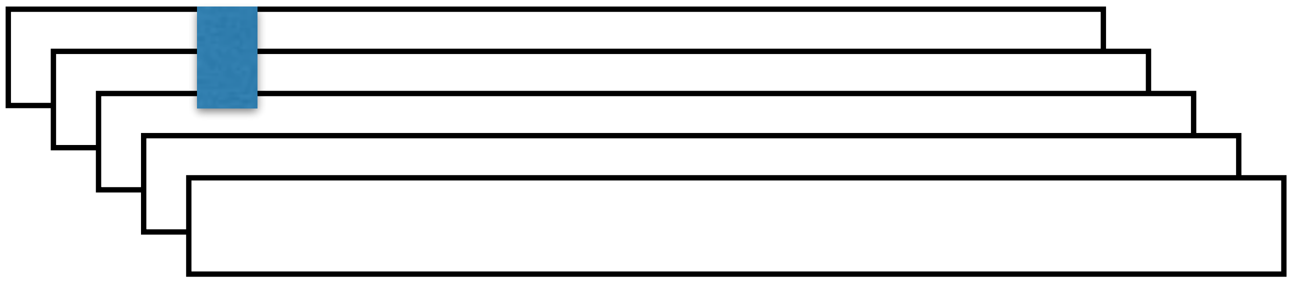
- ***Writing*** data becomes harder: append-only
  - No random writes
  - No overwrites
  - Must garbage collect to reclaim space

No Random Writes



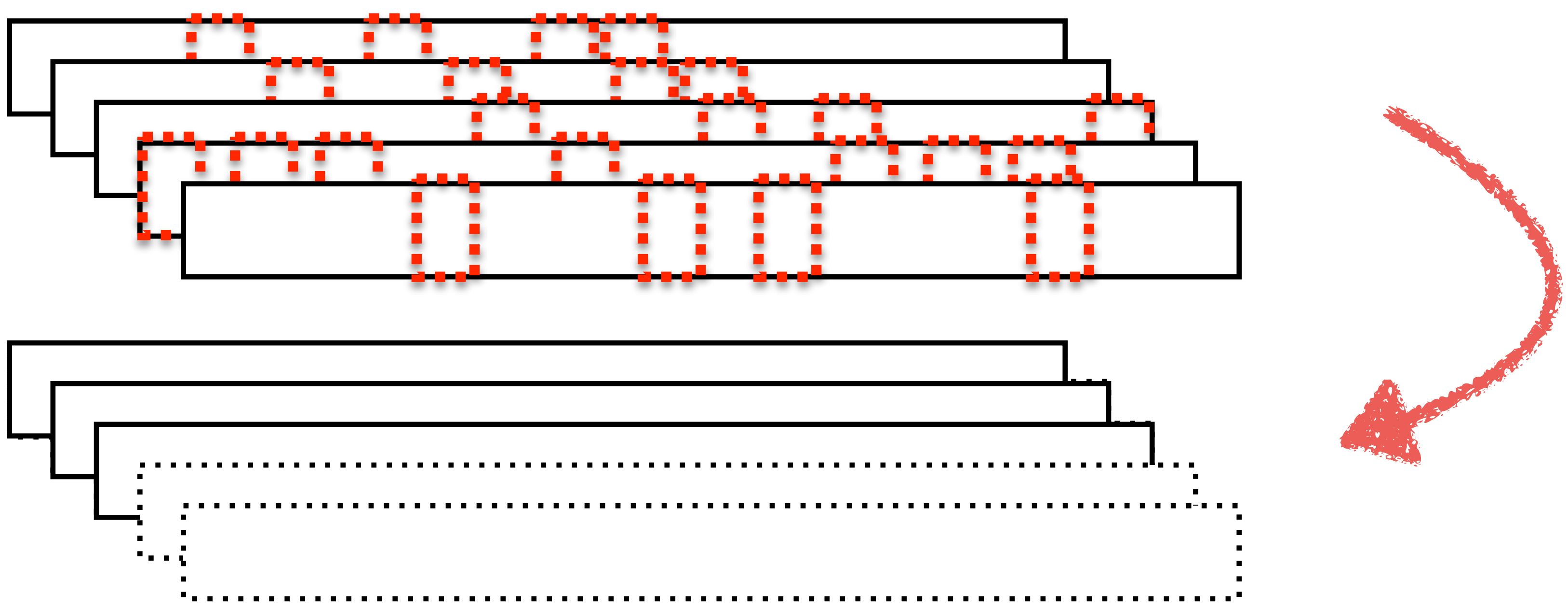
If we don't write to zones **append-only**, we could lose data

No Overwrites



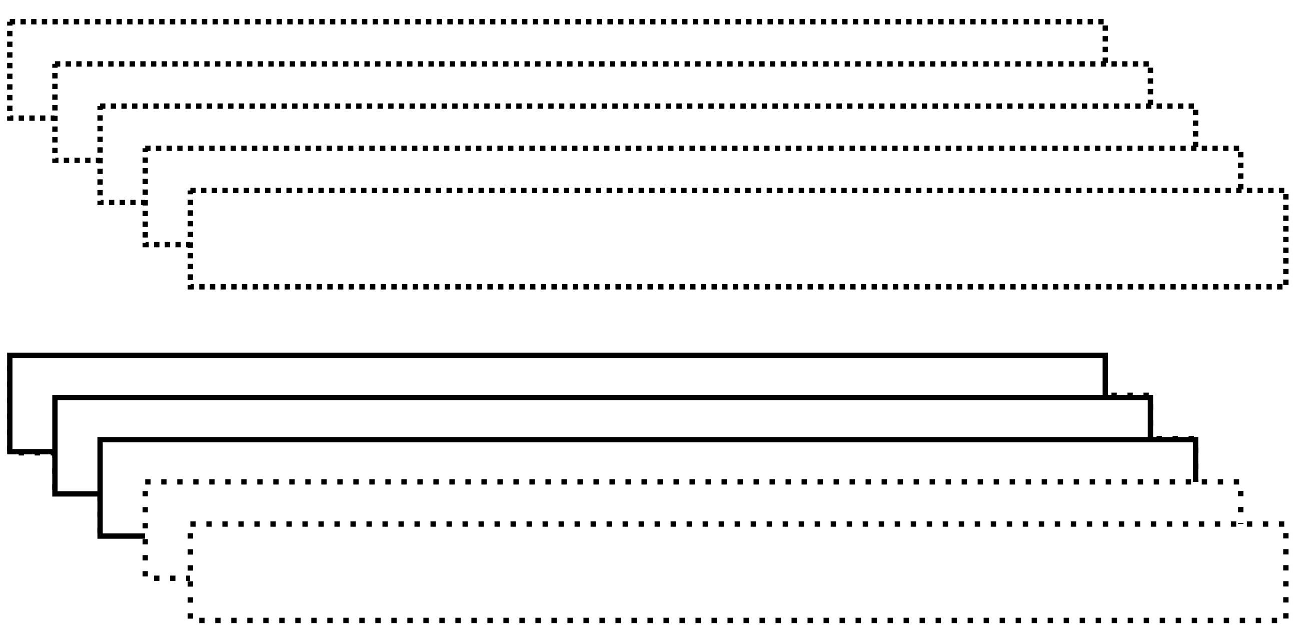
Must perform **out-of-place updates**, or suffer a read-modify-write of entire zone

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

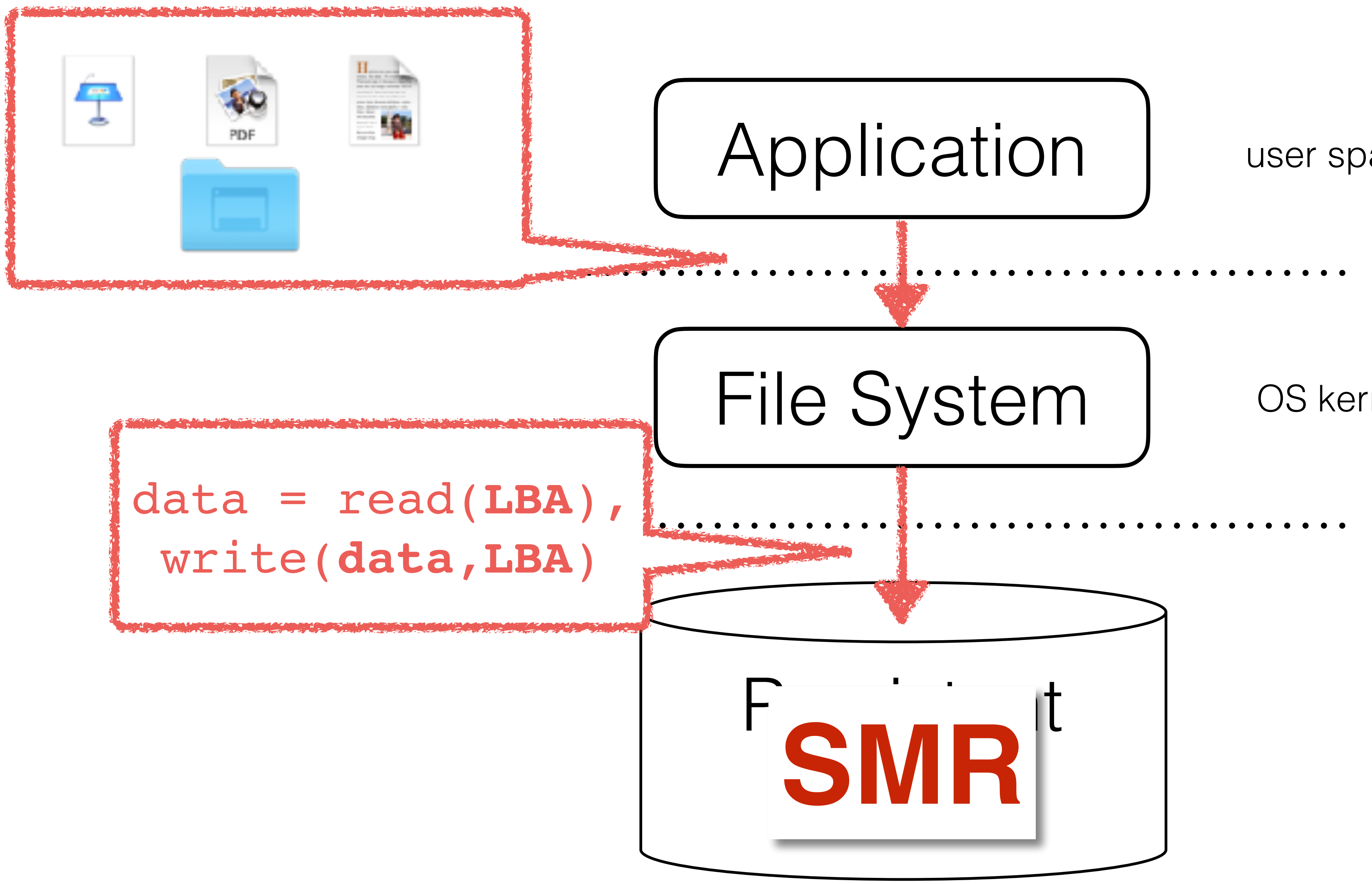
Recall HDD Observations

- **Problem**: Seeking is slow
- **Solution**: perform large sequential I/Os

Takeaway: HDD *performance optimizations* translate into SMR *correctness requirements*

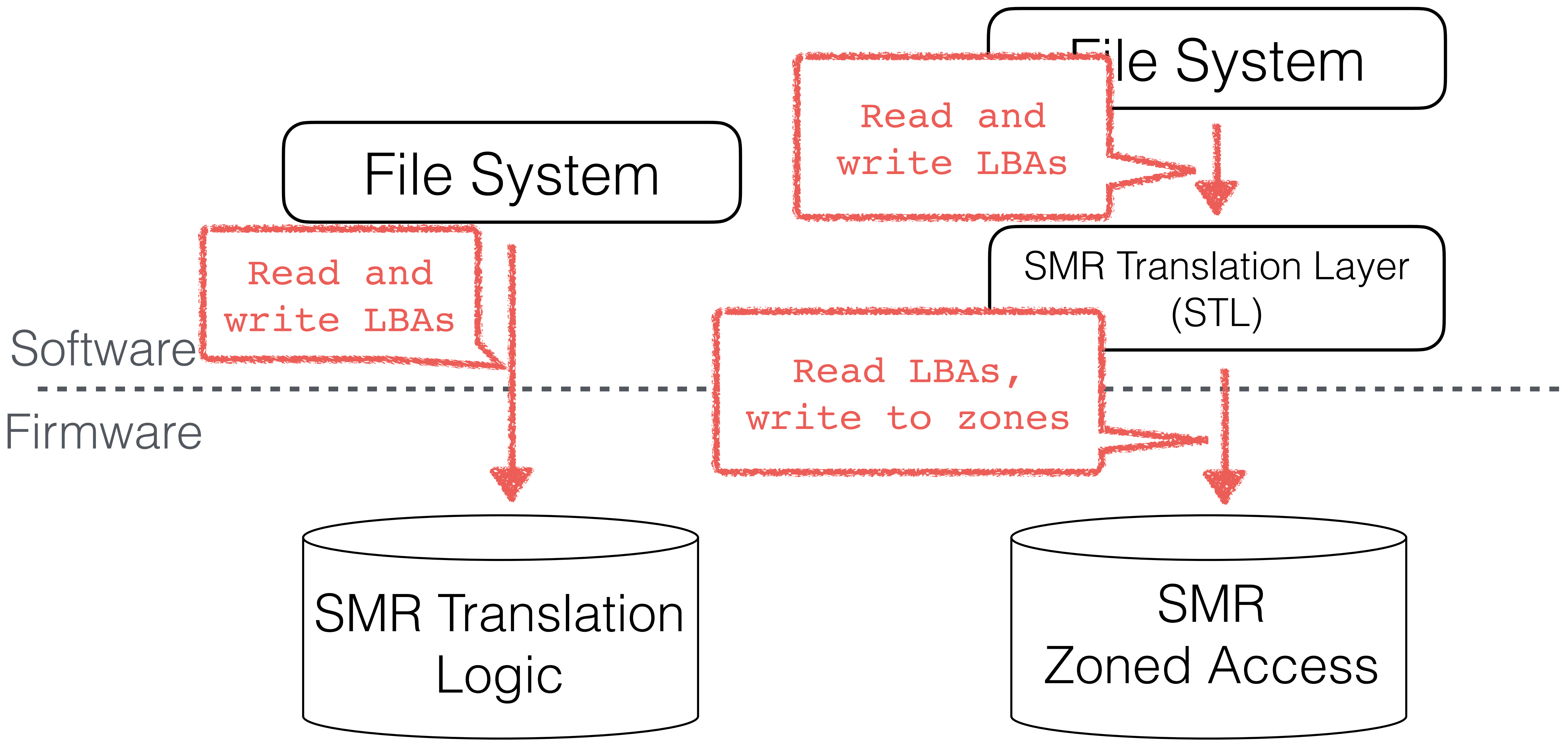
Implementing SMR Logic

Simplified Storage Stack



Question: who enforces the SMR write constraints?

Drive Managed vs. Host Managed



+ Easy to Deploy
- Limited HW resources

+ Flexible (more information)
- 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

[Hwang ’16 Transactions on Magnetics]

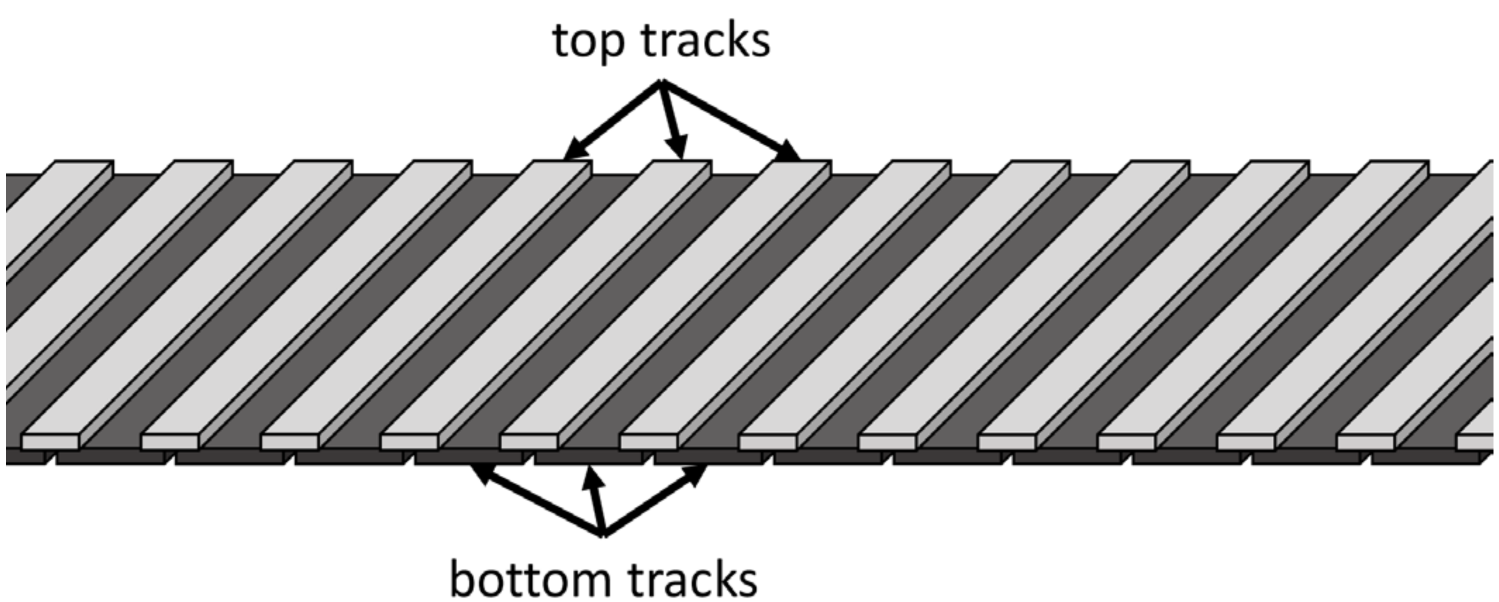


Figure 3: Depiction of interlaced track recording

[Feldman ’18 ;login:]

- 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

Magnetic Recording

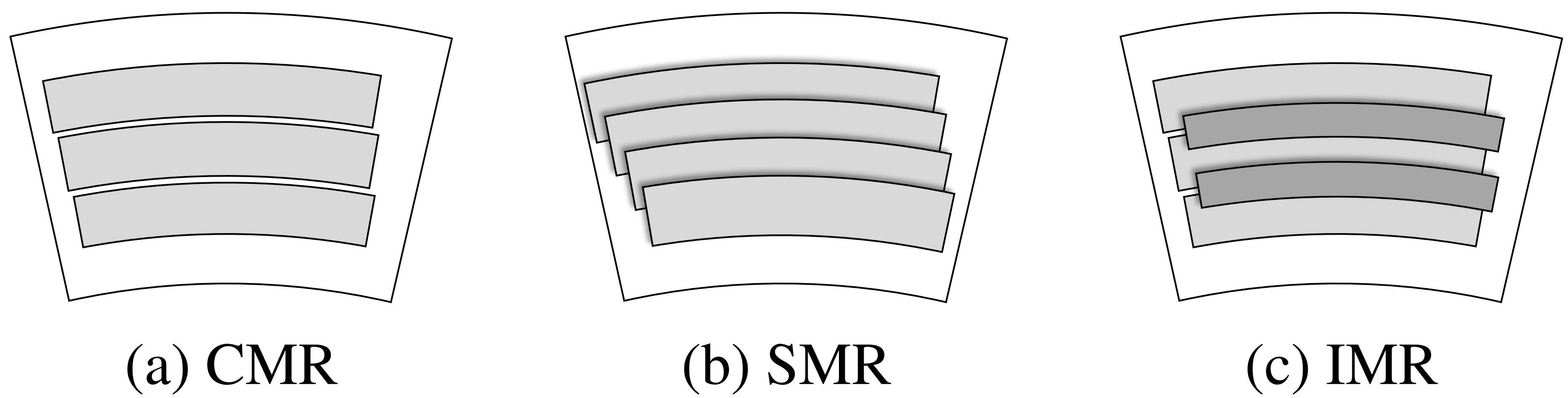


Figure 1: Track layout for CMR, SMR, and IMR.

[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