Track-based Translation layers for Interlaced Magnetic Recording (IMR)

MOHAMMAD H. HAJKAZEMI, AJAY KULKARNI, PETER DESNOYERS, TIMOTHY FELDMAN

July 2019
Outline

• What is Interlaced Magnetic recording (IMR)?
• Why does it need a translation layer?
• What are our proposals?
• How do they perform?
A quick disk overview

**Sector**
size = 512 - 4096 bytes

**Track**
size = 1~2MB

A 20 TB drive would have roughly about 13M tracks

1 rotation = 8~10ms
Magnetic recording technologies

- Tracks overlap
- **25% higher capacity** than CMR
- Available commercially for 5 years
- No in-place updates allowed
- Slower than CMR

- Tracks overlap
- 40% higher capacity than CMR
- Not commercially available
- Partially in-place updates allowed
- Can be faster than SMR
Magnetic recording technologies

- **Conventional magnetic recording (CMR)**
  - Tracks overlap
  - 25% higher capacity than CMR
  - Available commercially for 5 years
  - No in-place updates allowed
  - Slower than CMR

- **Shingled magnetic recording (SMR)**
  - Tracks overlap
  - 40% highest capacity than CMR
  - Not commercially available
  - Partially in-place updates allowed
  - Can be faster than SMR

- **Interlaced magnetic recording (IMR)**
  - Tracks overlap
  - No in-place updates allowed

How well can IMR perform?
Interlaced magnetic recording

• Half of the tracks overlap
  • Bottom tracks are overlapped by top tracks
• Top tracks are narrower
  • Hold 80% -90% as much data
• No in-place updates are allowed for bottom tracks
  • Solution : RMW or using a translation layer
SMR and IMR translation layers

• Goal: provide conventional block interface
• SMR drive based on translation layer location
  • Host-managed
  • Drive-managed
• IMR
  • Our focus is on drive-managed IMR
IMR top/bottom track update

Top track update operation

- Write head
- Read head
IMR top/bottom track update

Top track update operation

Bottom tracks still could be read if top tracks are updated
IMR top/bottom track update

Bottom tracks still could be read if top tracks are updated
IMR top/bottom track update

Top track update operation

Bottom track update operation

Due to bottom track update, top track data is **corrupted** and therefore cannot be read.

Bottom tracks still could be read if top tracks are updated.
Read-modify-write: a simple translation layer

Update $S_T$:

1. Read $S_{T-1}$ and $S_{T+1}$ and make copies
2. Update $S_T$
Read-modify-write: a simple solution

**Update \( S_T \):**

1. Read \( S_{T-1} \) and \( S_{T+1} \) and make copies
2. Update \( S_T \)
3. Write back \( S_{T-1} \) and \( S_{T+1} \)

RMW imposed 2 and 3 additional reads and writes for a single update.
RMW performance

- Synchronous

- Overhead per bottom track update:
  - Short writes
    - \( \text{RMW Latency} \approx t_{\text{seek}} + 4 \times t_{\text{rotation}} + t_{\text{transfer}} \)
  - Large writes
    - \( \text{RMW Latency} \approx t_{\text{seek}} + 5 \times t_{\text{rotation}} + t_{\text{transfer}} \)
  - Poor performance compared to CMR
    - \( \text{Conventional drive latency} \approx t_{\text{seek}} + \frac{1}{2} t_{\text{rotation}} \)
Improving RMW

- Our Strategy: Get the hot data out of bottom tracks
  - Minimize RMW operation

- But what granularity?
  - Per sector?
    - Too much memory to keep the sector map
    - Fragmentation and large number of seeks
  - Single track maybe?
Track access pattern and locality

A small number of tracks receives majority of writes
Proposed IMR track-based translation layers

- Algorithms
  - Track flipping
  - Selective track caching
  - Dynamic track mapping
- Runs periodically (e.g., every 20K write operations = every few minutes) and in the background
- Limited number of tracks remapped every iteration
  - Limited performance overhead
- Still need RMW
Track flipping

- **Hot bottom tracks** are swapped with neighboring **cold top tracks**

- **Challenges and limitations:**
  - Differing top/bottom track sizes
  - Solution: move either low or high LBAs, whichever is hotter
  - No improvement if both neighboring top tracks are hot as well

\[
\text{cost} \approx 3t_{\text{seek}} + 8t_{\text{rotation}}
\]
Track flipping – memory requirement

- Hot track detection
  - logging the written track
  - Less than 0.25 MB

- Track map
  - 5 states for each bottom track (Non-flipped, 4 flipped states)
  - 3 bits per two tracks
  - 2.5 MB for a 20T drive
Selective track caching

- Hot bottom tracks are cached in a small non-interlaced reserved area
- Hot bottom tracks are promoted to the cache
- Cold tracks are demoted to their home locations
- Addresses the limitations of track flipping
Selective track caching

- Hot bottom tracks are cached in a small non-interlaced reserved area
- Hot bottom tracks are promoted to the cache
- Cold tracks are demoted to their home locations
- Addresses the limitations of track flipping

\[\text{evict} + \text{insert} \approx 6t_{\text{seek}} + 10t_{\text{rotation}}\]
Selective track caching - memory requirement

- Hot track detection
  - logging the written track
    - Less than 0.25 MB

- Look-aside cache map
  - Proportional to the number of tracks in cache
  - Tiny (for 100 track cache in our experiments)
Dynamic track mapping

- Arbitrary permutation of tracks within zones
- Concatenate all LBAs and group them in fixed size pseudo-tracks
- Requires about 12.5 MB for a 20TB drive with zone size of 256 tracks
- Requires 0.25 MB for hot pseudo-track detection
- Addresses the limitation of track flipping

\[
\text{swap 2 tracks} \approx 5t_{\text{seek}} + 13t_{\text{rotation}}
\]
Simulation setup: traces and disk

- CloudPhysics traces
  - Block traces from VMs running Linux and Windows
  - LBA range of 10s of GBs to 1.5 TB
  - Very short inter-arrival time

- Disk Model
  - 6K rpm disk
  - Ignore head switch
  - Rotational delay = ½ plater revolution
  - Seek time : 2ms to 20ms LBA range dependent
  - Track size = 2MB for both top and bottom tracks
  - Write cache enabled
Simulation setup: I/O latency model

- IO latency includes:
  - Host and device queuing
  - Depth of 64
  - Seek time
  - Rotational delay
  - Transfer time

\[ Latency \approx t_{\text{queuing}} + t_{\text{seek}} + \frac{1}{2} t_{\text{rotation}} + t_{\text{transfer}} \]
Results: write amplification factor

High WAF due to RMW

Proposed translation layers reduce WAF
Results: normalized mean latency

Latency normalized to CMR

- **w08**
- **w26**
- **w39**
- **w46**
- **w56**
- **w84**

Workload:
- **Read dominant**
- **Write dominant**
- **High inter-arrival time**

Latency normalized to CMR:
- **RMW**
- **Flipping**
- **Caching**
- **Dynamic mapping**
Summary

- Interlaced magnetic recording
  - Half of the tracks overlap
  - Higher capacity compared to conventional and shingled drives
  - Relaxed constraints relative to SMR
- Read-modify-write is a solution
  - Poor performance
- Proposed alternatives translation layers
  - Track flipping
  - Track caching
  - Dynamic track mapping
  - Take advantage of the IMR flexibility
  - Improve the performance significantly
Summary

• Interlaced magnetic recording
  • Half of the tracks overlap
  • Higher capacity compared to conventional and shingled drives
  • Relaxed constraints relative to SMR
• Read-modify-write is a solution
  • Poor performance
• Proposed alternatives translation layers
  • Track flipping
  • Track caching
  • Dynamic track mapping
  • Take advantage of the IMR flexibility
  • Improve the performance significantly

Questions?