SMaRT: An Approach to Shingled Magnetic Recording Translation

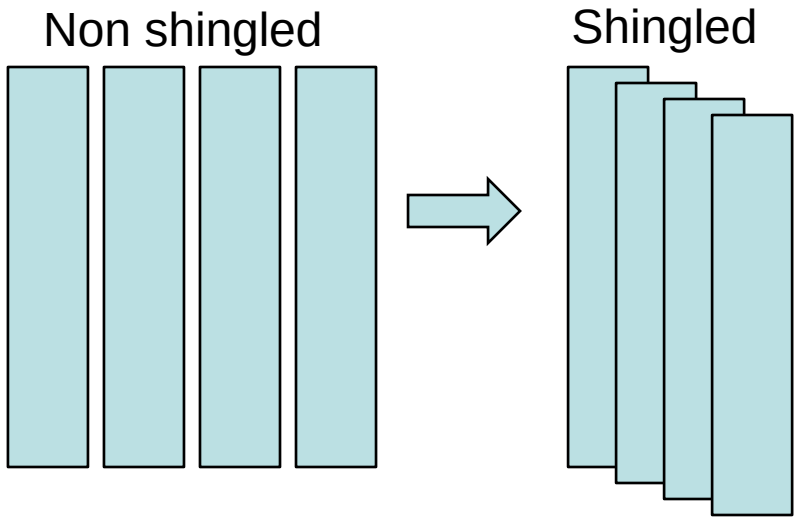
Weiping He and David H.C. Du

Outline

- SMR Backgrounds
  - Characteristics
  - Types of SMR Drives
- Challenges
  - Write Amplification
  - GC Overhead
- Motivations
  - Current Design of SMR drives
  - Advantages of Track-Based Mapping
- Proposed Approach
  - SMaRT
- Evaluations

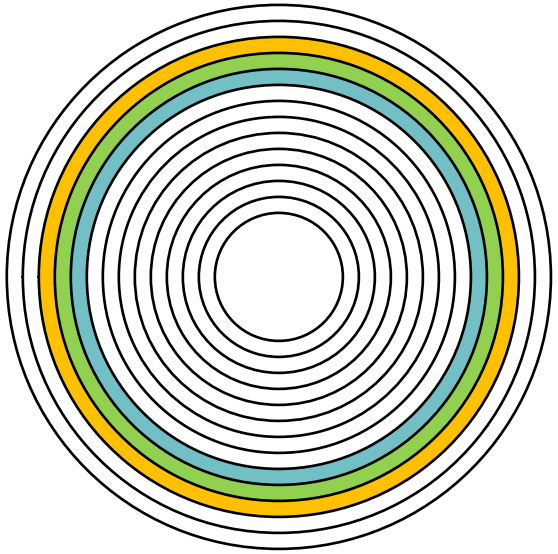
SMR Background

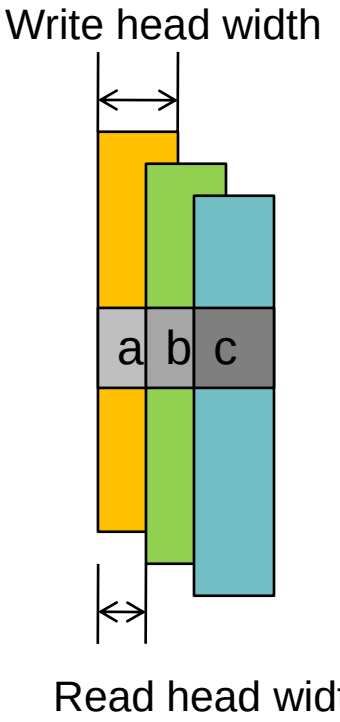
- Traditional HDDs (perpendicular magnetic recording) are reaching areal density limit
- Shingled magnetic recording (SMR) is a new promising technology by overlapping tracks

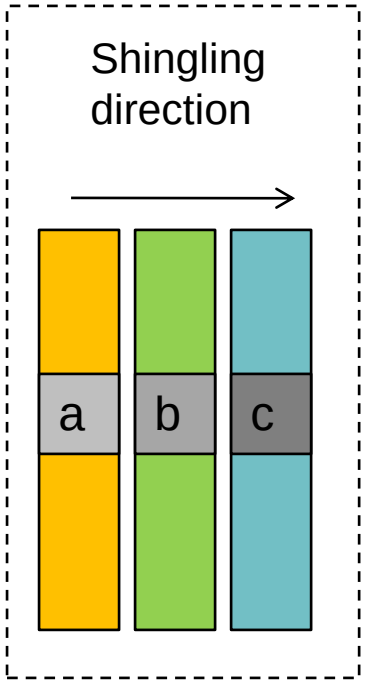


SMR Characteristics

- Write head width is larger than read head width
- Write/update a block *in place* may destroy the valid data on the subsequent tracks if any
- Sequential write is preferred







Simplified diagram

Current Types of SMR Drives

- Device-Managed SMR (DM-SMR)
  - The device handles address mapping
  - Block I/O interface
  - Drop-in replacement for HDDs.
  - E.g., Seagate 8TB Archive [1]
- Host-Aware SMR (HA-SMR) – T10 and T13
  - The host is preferred to follow I/O rules (e.g., writing data sequentially to the location of write-pointer in each zone).
  - I/Os violating the rules will be processed in a DM-SMR way. i.e., go to persistent cache.
- Host-Managed SMR (HM-SMR) – T10 and T13
  - The host has to strictly follow rules
  - I/Os violating the rules will be rejected.
  - E.g., WD/HGST 10TB UltraStar Ha10 [2]

Current Types of SMR Drives

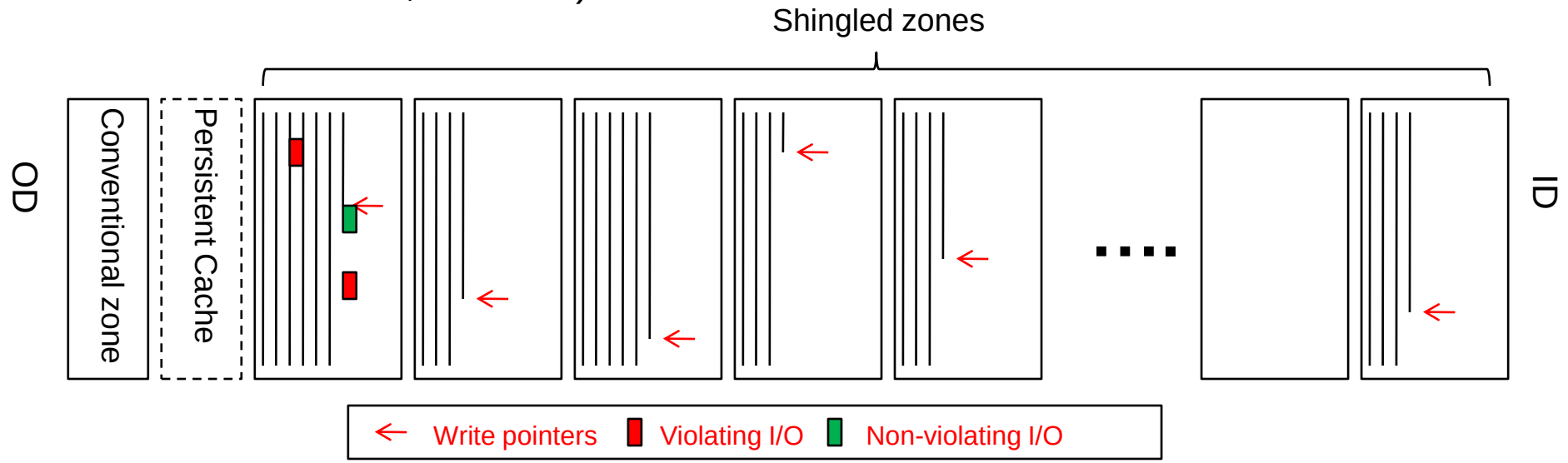
Zone Configurations

	DM-SMR	HA-SMR	HM-SMR
Conventional zone	**Mandatory**	Optional	Optional
Persistent Cache	Optional	Optional	Optional
Seq. write pref. zone	Not supported	**Mandatory**	Not supported
Seq. write req. zone	Not supported	Not supported	**Mandatory**

More Information on HA-SMR and HM-SMR can be referred to a presentation by Tim Feldmann
- Host-Aware SMR (Tim Feldmann OpenZFS ‘14) [3]

Basic Layout of SMR Drives

- Conventional Zones
  - Miscellaneous usages: metadata, journal, etc.
- Shingled Zones
  - DM-SMR: Present a consecutive logical space to host
  - HM-SMR: sequential write **required** zones (fail violating I/Os)
  - HA-SMR: Sequential write **preferred** zones (direct violating I/Os to cache, GC later)

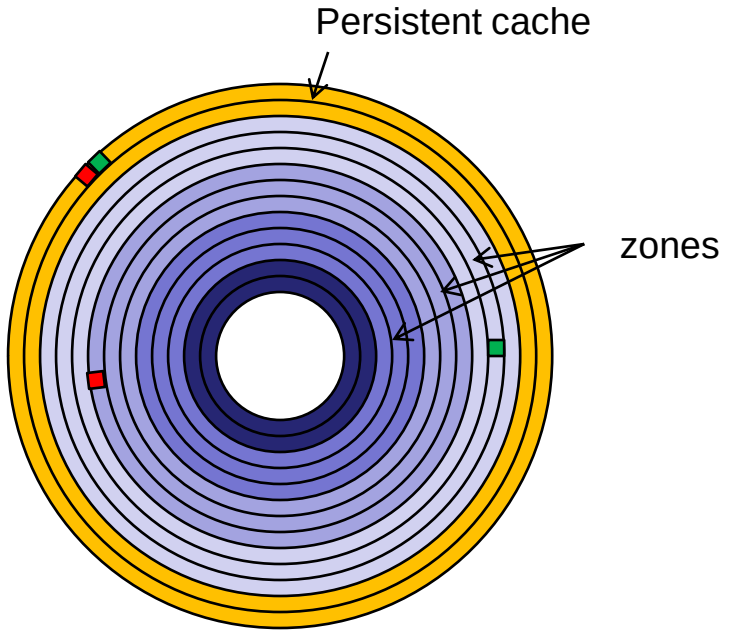


Current Types of SMR Drives

- Device-Managed SMR (DM-SMR)
  - The device handles address mapping
  - Block I/O interface
  - Drop-in replacement for HDDs.
  - E.g., Seagate 8TB Archive [1]
- Host-Aware SMR (HA-SMR) – T10 and T13
  - The host is preferred to follow I/O rules (writing data sequentially to the location of write-pointer in each zone).
  - I/Os violating the rules will be processed in a DM-SMR way. i.e., go to persistent cache.
- Host-Managed SMR (HM-SMR) – T10 and T13
  - The host has to strictly follow rules
  - I/Os violating the rules will be rejected.
  - E.g., WD/HGST 10TB UltraStar Ha10 [2]

Challenges

- Challenges of DM-SMRs:
  - Write amplifications (one write becomes multiple writes)
  - Garbage collections (persistent cache cleaning and zone cleaning)
- One of Seagate's Solutions [4]:
  - Persistent cache
  - Static mapping for zones.
  - Aggressive GCs
- Pros:
  - Simple and clean
- Cons:
  - Workload picky: Suitable for workloads with idle times.
  - Data staging in persistent cache



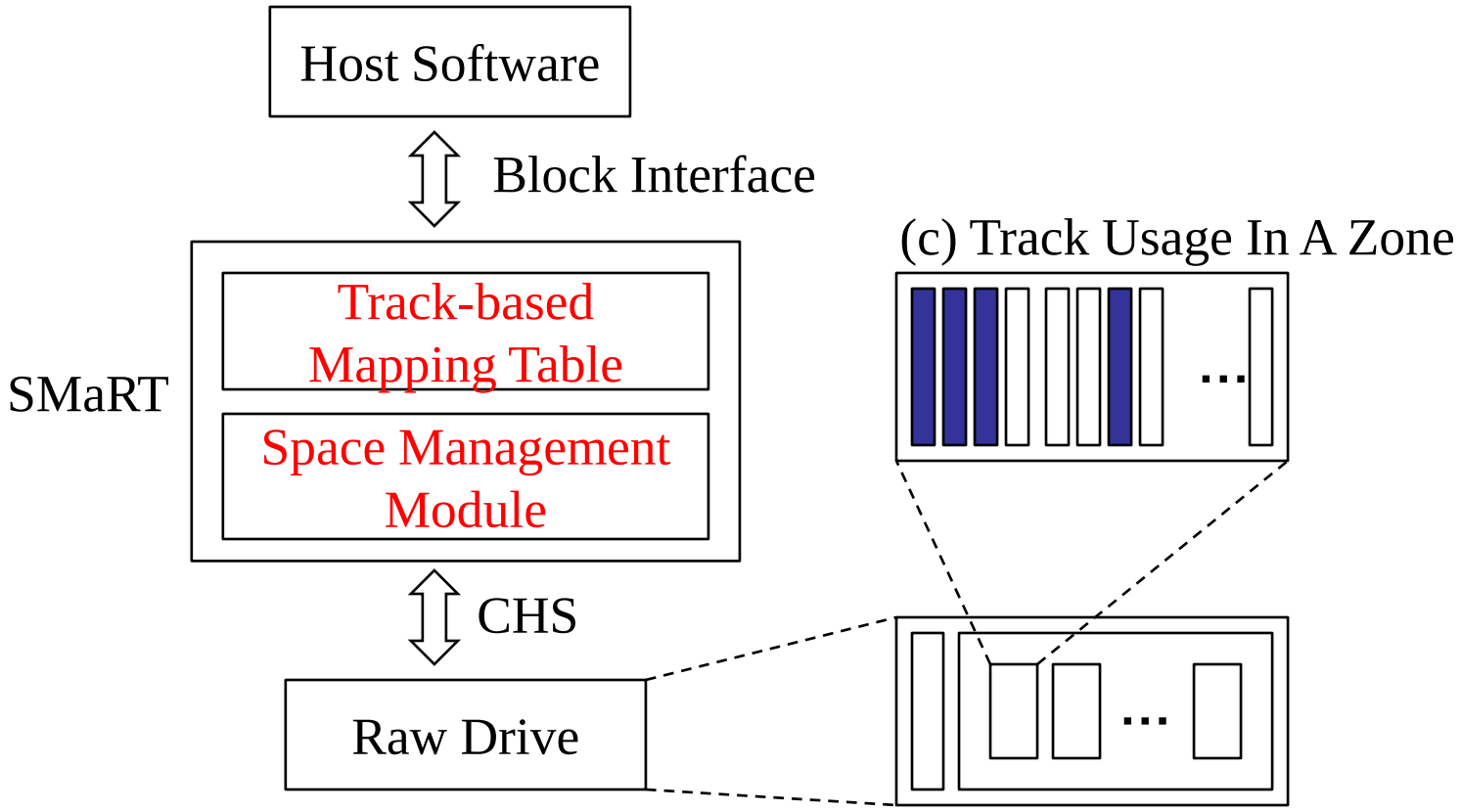
Motivations

- Two inherent properties of SMRs
  - Advantage of Track-based mapping
    - An invalid track can be reused immediately without “erase” like operations in SSDs
    - Block-based mapping will create huge mapping table and will introduce “invalidated” blocks problem (cannot be used right away)
  - A track supports in-place update if its following track is free.
- Can we exploit these properties to … ?
  - Reduce write amplification
  - Reduce read fragmentations
  - Improve overall I/O performance
  - Remove or mitigate the use of persistent cache

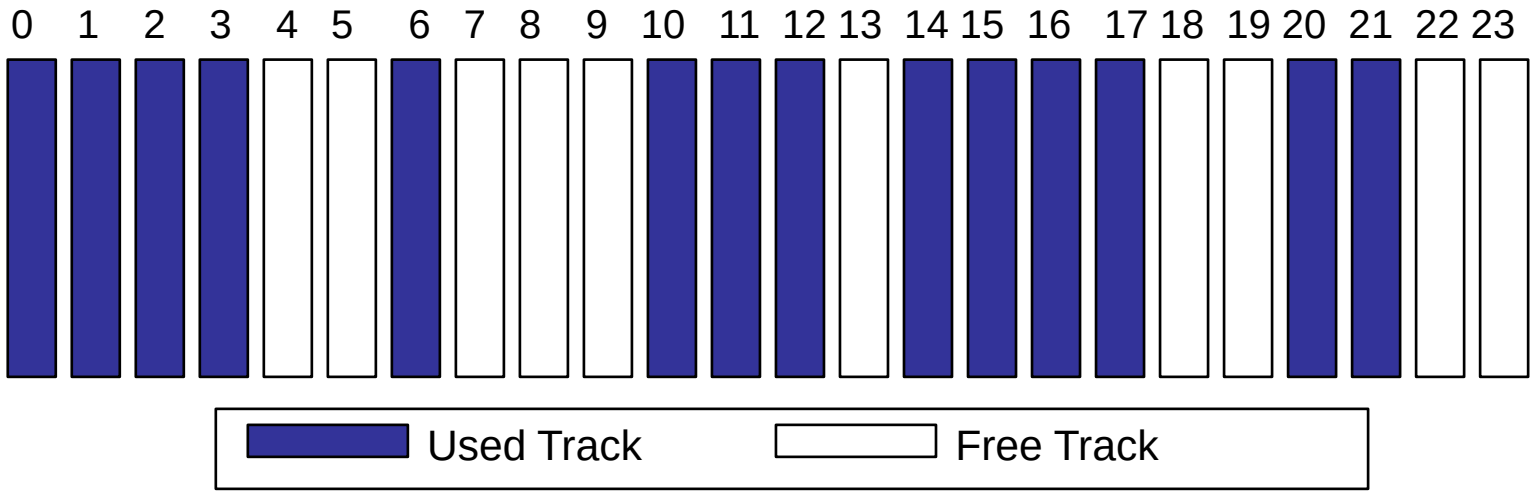
Our Proposed Solution: SMaRT

- SMR Drive Layout Assumption
  - Conventional zone
  - Many Shingled zones
- Two Function Modules Are Designed:
  - A dynamic track-based mapping table
    - It supports block-level address mapping
    - Hybrid update strategy
  - A space management module which handles
    - free track allocation, AND
    - Garbage collection

SMaRT Overall Architecture



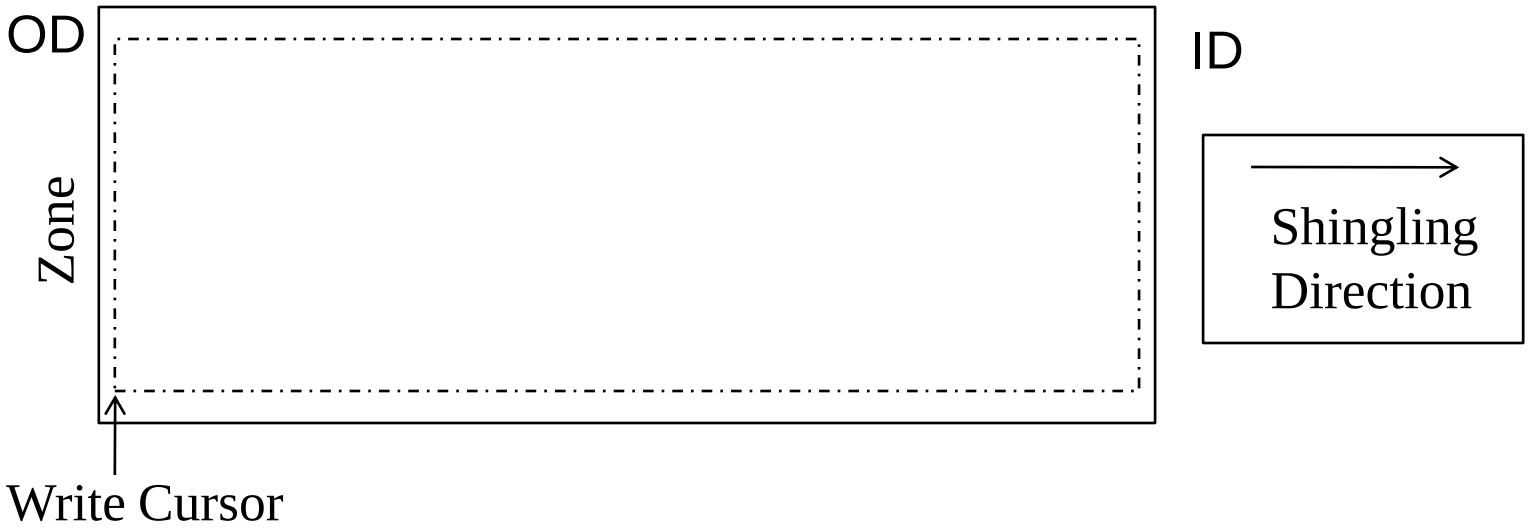
SMaRT Space Management – 1
Space, Space Element and Hybrid Update



- **Free space**: [[4, 5], [7,8,9], [13], [18, 19], [22, 23]]
  - Free space element: a group of consecutive free tracks.
  - Tracks 4, 7, 8, 18, 22 are usable
  - Bigger free space groups have more usable tracks.
- **Used space**: [[0, 1, 2, 3], [6], [10, 11, 12], [14, 15, 16, 17], [20, 21]]
  - Used space element: a group of consecutive used tracks.
  - Tracks 3, 6, 12, 17 and 21 support in-place update.

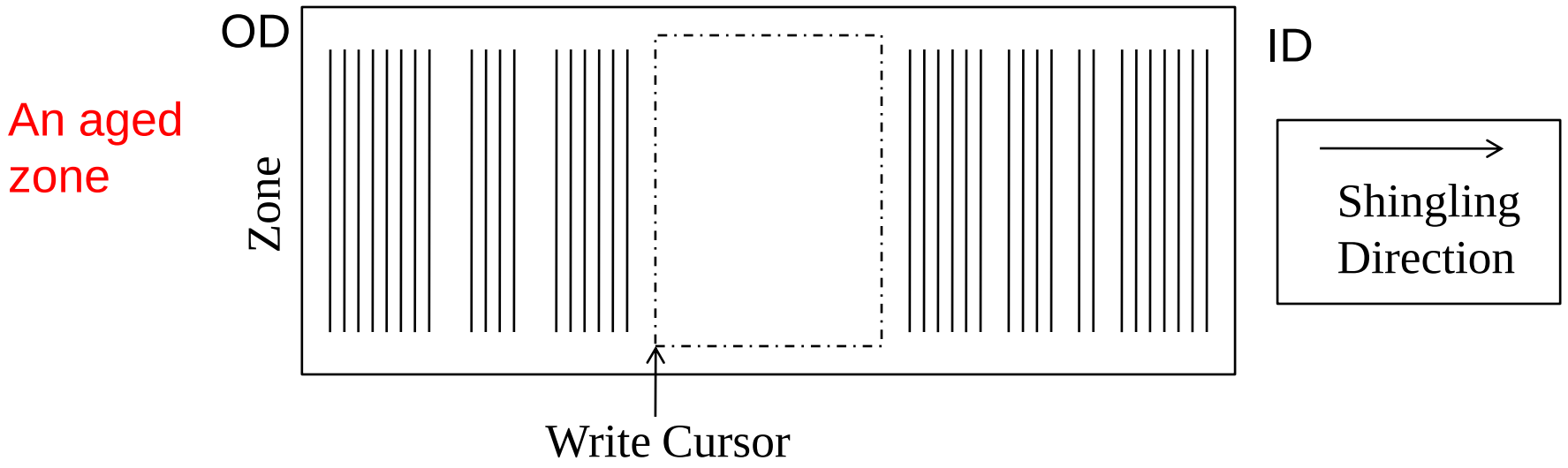
SMaRT Space Management – 2
Track Allocation

An empty zone



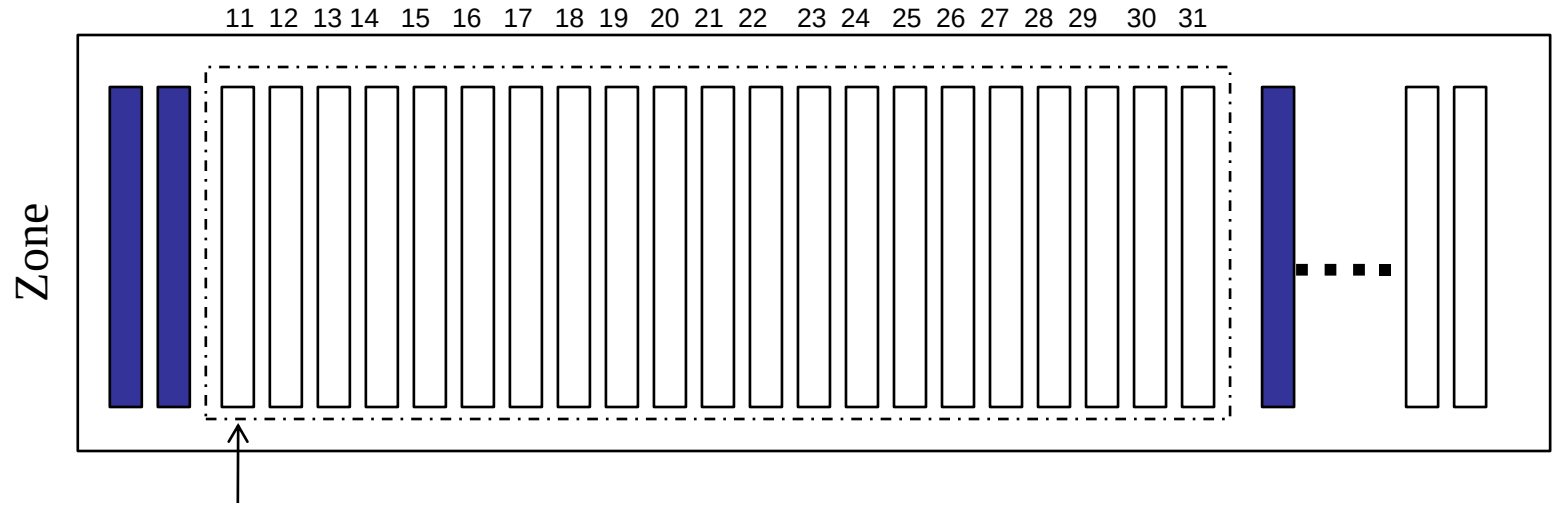
- **Allocation pool** is the largest free space element.
  - The whole zone is an allocation pool for an empty zone.
- **Write cursor** is used to indicate the next available free track for data allocation.

SMaRT Space Management – 2
Track Allocation



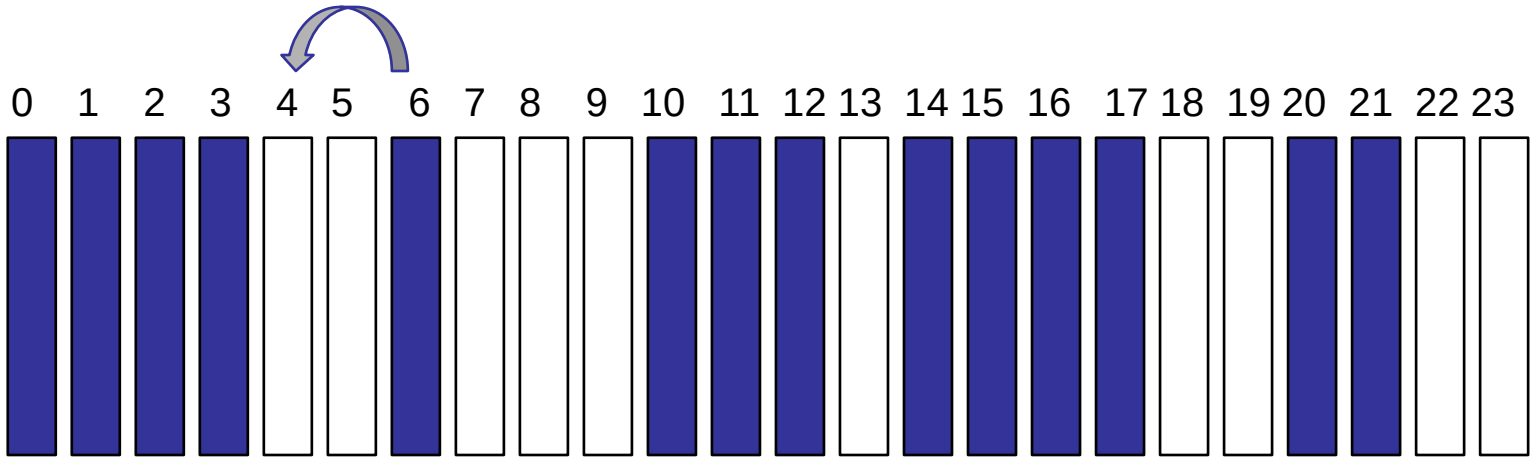
- **Allocation pool** is the largest free space element.
  - The whole zone is an allocation pool for an empty zone.
- **Write cursor** is used to indicate the next available free track for data allocation.

Track Allocation Example



- All writes (new data and updated data) go to the write cursor sequentially
- Newly updated tracks are deemed as hot
  - Hot tracks are predicted to be accessed again in the near future
- SMaRT allocates an extra track as safety gap for each hot track if space utilization is less than 50%.
- When the current allocation pool is fully consumed, choose the currently largest free space element as the new allocation pool.

SMaRT Space Management – 3
Garbage Collection



- Fragmentation Ratio R (Evaluated for incoming writes.)
  - F: total number of free tracks
  - N: number of free space elements

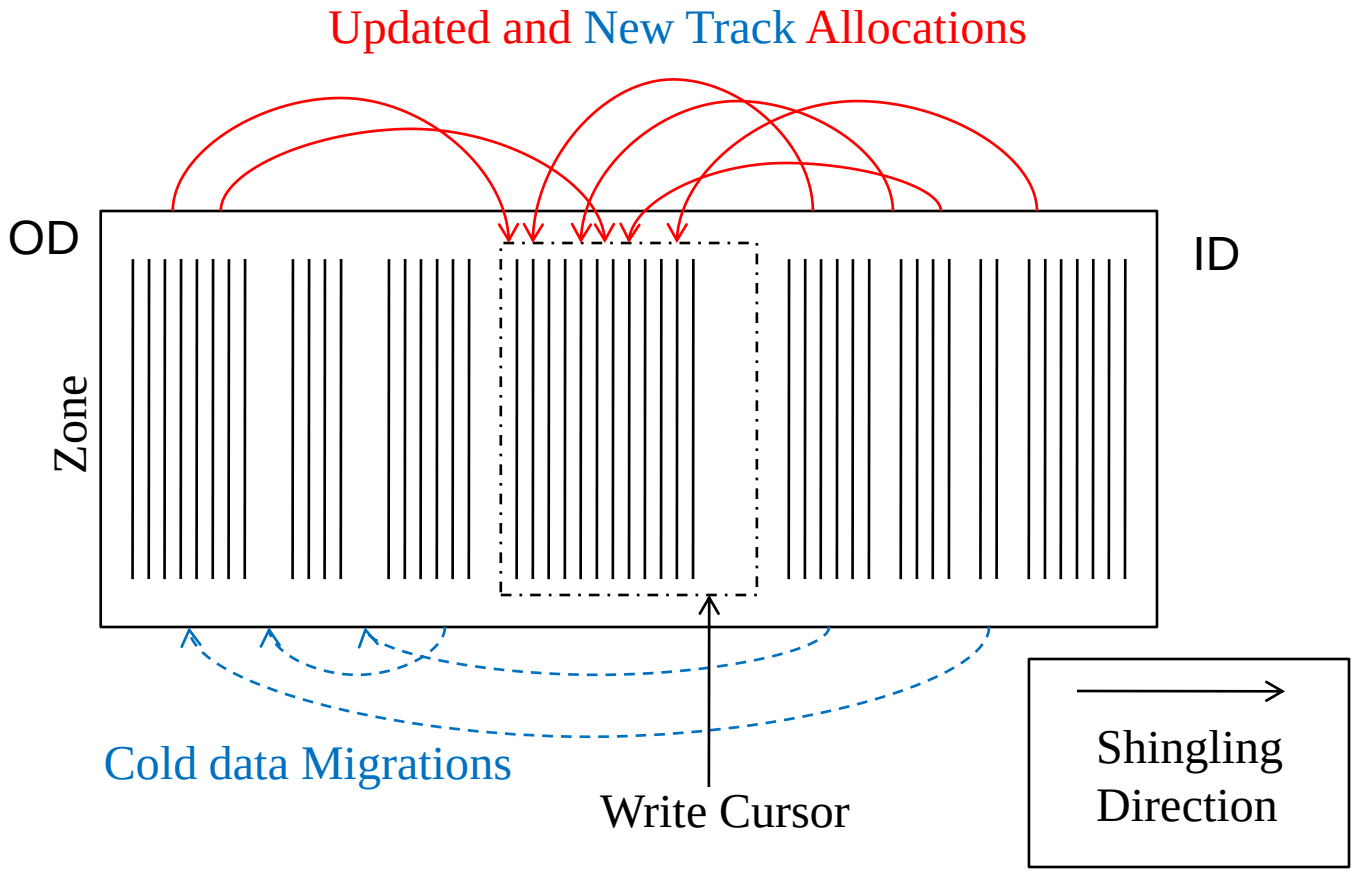
$$R = \frac{F - N}{F}, where\ 1 \leq N \leq F$$

- Pick victim
  - A small used space element of size W
  - U is the space utilization.

$$W = \frac{U}{1 - U}, where\ 0 < U < 1$$

- Pick destination
  - Allocate to the first free space element to the left that fits it.
  - Or simply shift left and append to its left neighbour if failing to meet the above condition.

SMaRT Space Management – 4
Automatic Cold Data Progression

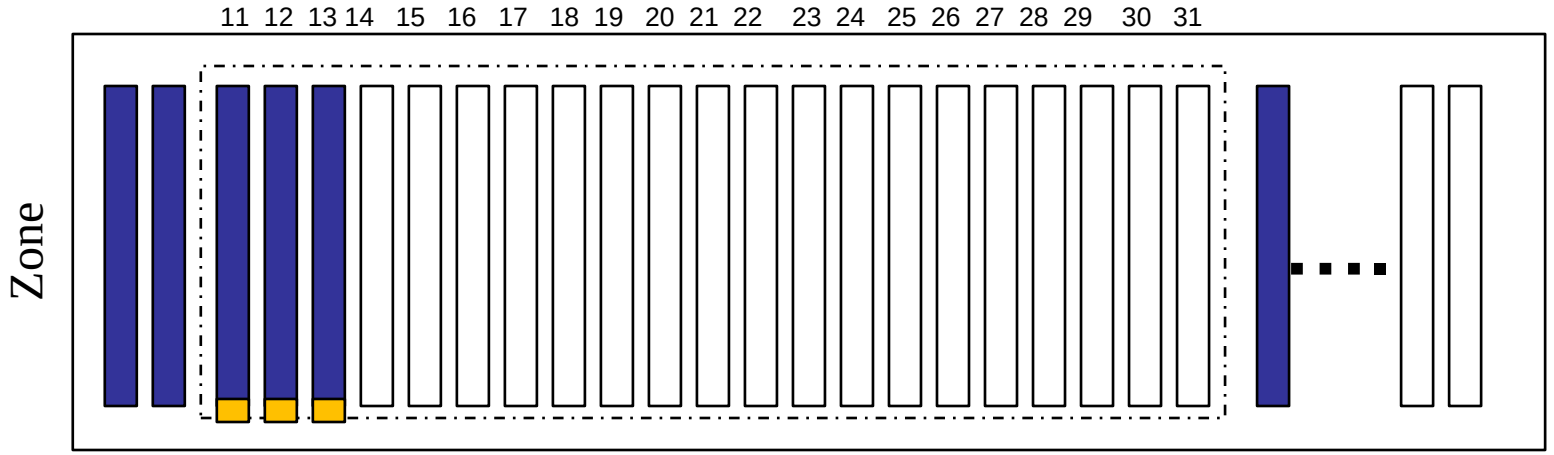


- GC is essentially free space consolidation
- GC algorithm
  - Pick victim
  - Pick appending destination
- Cold data migration
  - "hot" as recently updated data

Scheme Reliability

- Power failure can happen before the updates to the mapping table is flushed to disk.
- We designed an economic solution based on Backpointer-Assisted Lazy Indexing [5]
  - Store a backpointer to the logical track when writing a physical track
  - Flush mapping table whenever an allocation pool is fully consumed.
- To recover from power failure:
  - Scan the latest allocation pool
  - Append these LTN-to-PTN mapping entries to the disk copy

Scheme Reliability



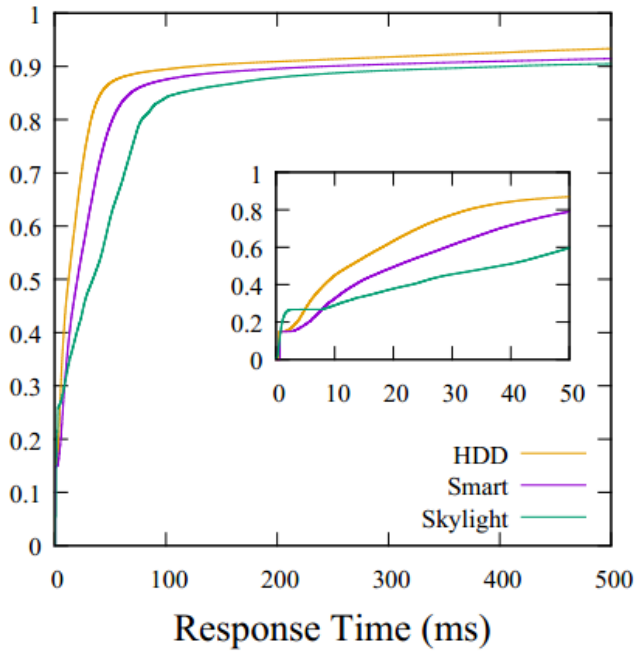
Timestamp	PTN	LTN
T1	11	X1
T2	12	X2
T3	13	X3
T – y1	14	X4
T – y2	15	X5

+ Mapping table on disk = Recovered Mapping table

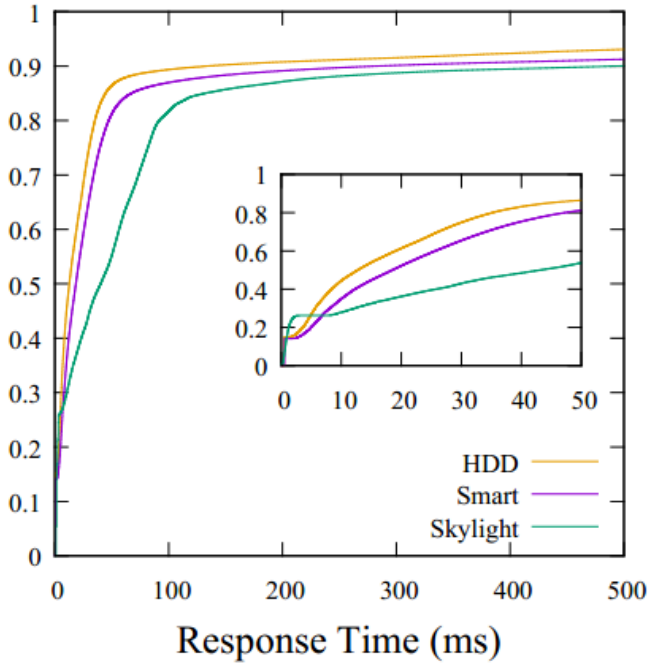
Evaluations

- Competitor schemes:
  - HDD
  - Seagate SMR drive exploited in Skylight (denoted as “Skylight”)
- Trace-based simulations:
  - Seagate Cheetah disk drive
    - 146GB based on 512B block or 1.1TB based on 4KB block
- Traces:
  - mds_0, proj_0, stg_0 and **rsrch_0**
  - Write intensive
- Evaluation points for drive utilizations:
  - 30%, 60% and 90%
- Measure Metrics:
  - Response time, read fragmentation, write amplifications and GC overhead

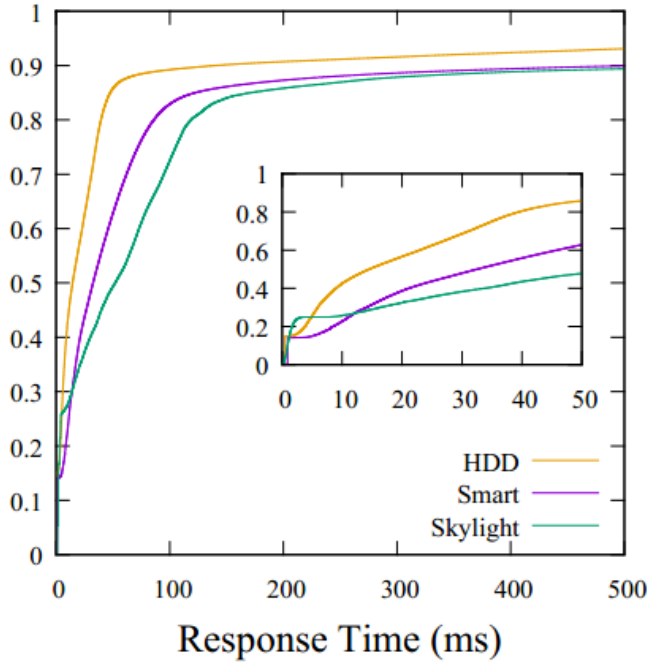
Response Time



rsrch_0 @ 30%



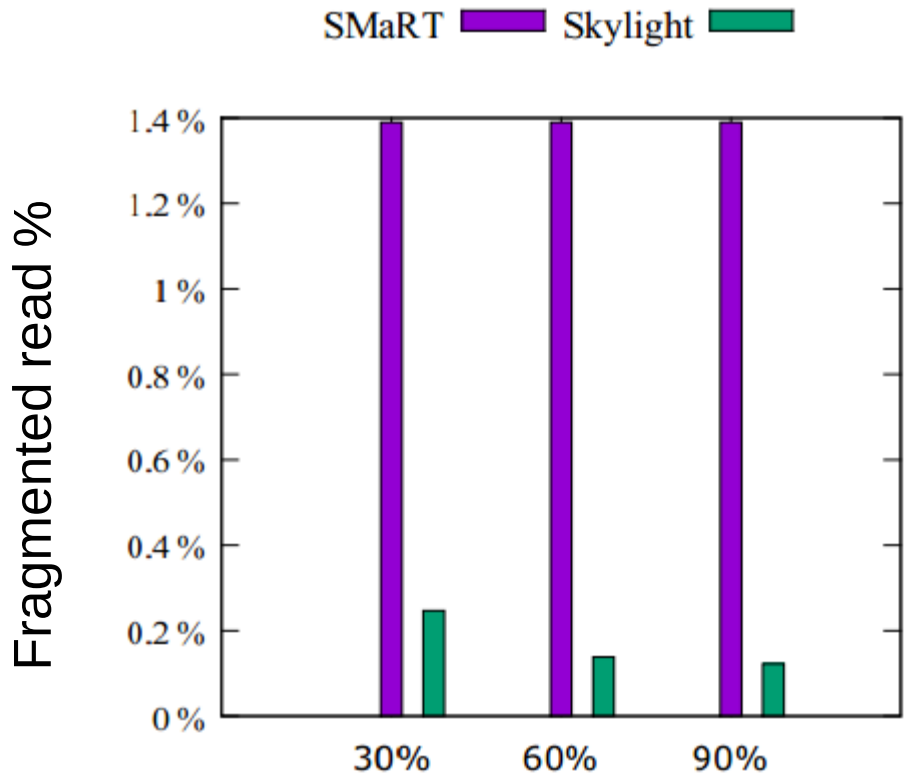
rsrch_0@60%



rsrch_0@90%

Response time: the difference between the time a request is queued and the time it is completed.
Skylight briefly crosses HDD and SMaRT in the lower range, due to persistent cache.
Skylight lags behind for the majority of the requests and response times.

Read Fragmentation - 1

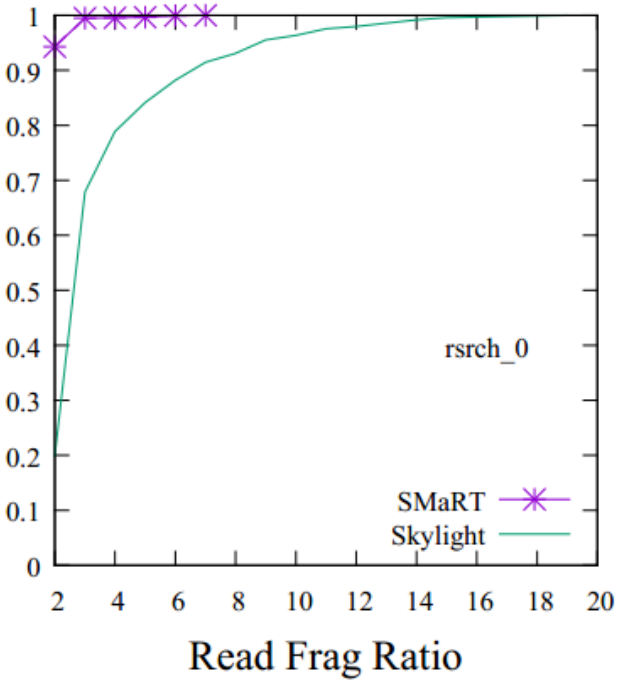


The percentage of fragmented reads.

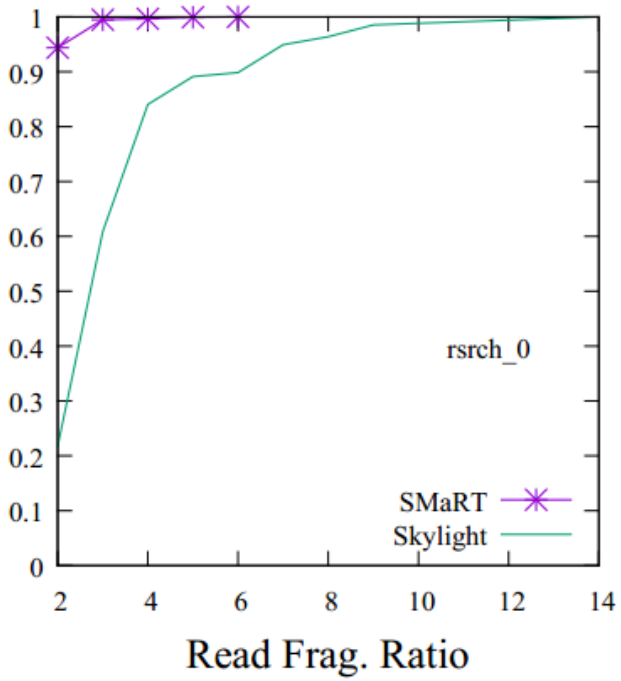
SMaRT is more consistent because it's mostly decided by the request sizes.

Skylight is a bit more random, depending on how data scatters between persistent cache and zones.

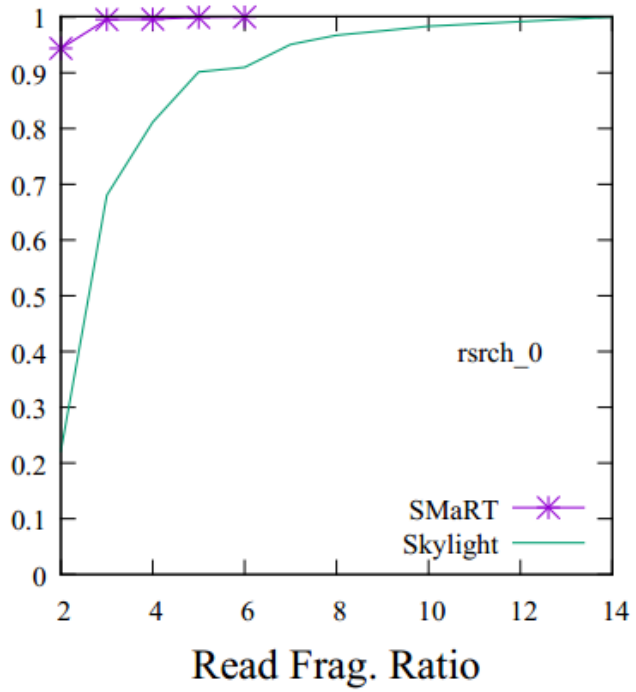
Read Fragmentation - 2



rsrch_0 @ 30%



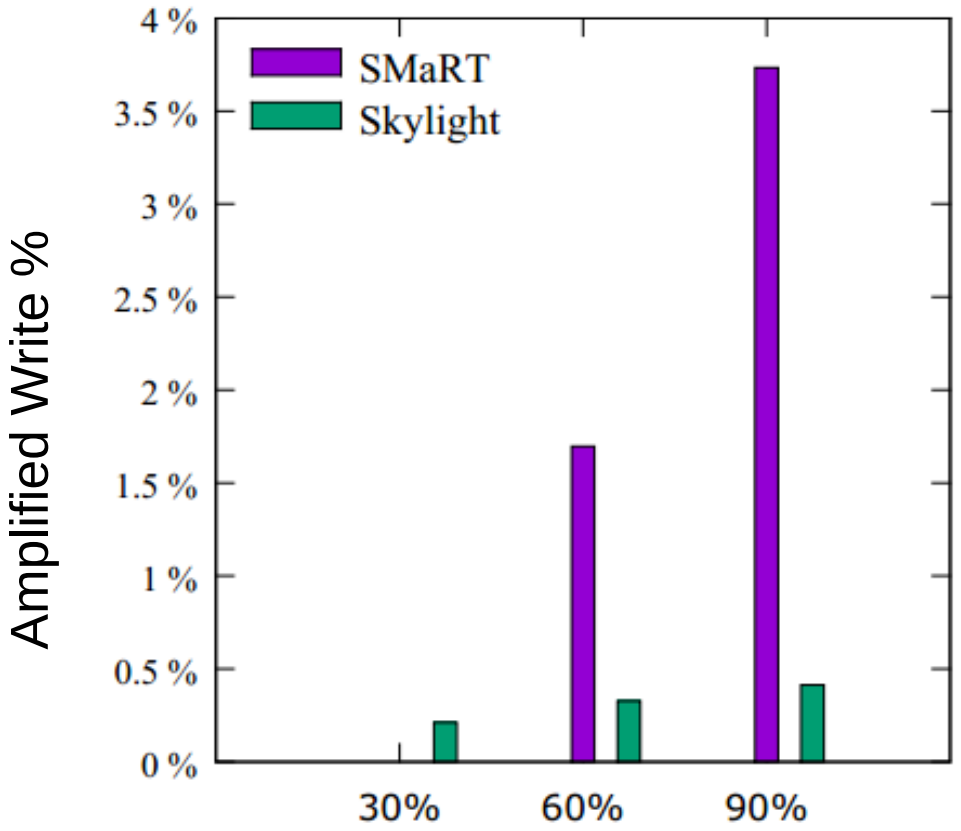
rsrch_0@60%



rsrch_0@90%

Read fragmentation ratio: the number of sub-reads created by a single read request.
SMaRT is more consistent and has narrower spectrum.
Skylight has wider specturm

Write Amplification - 1

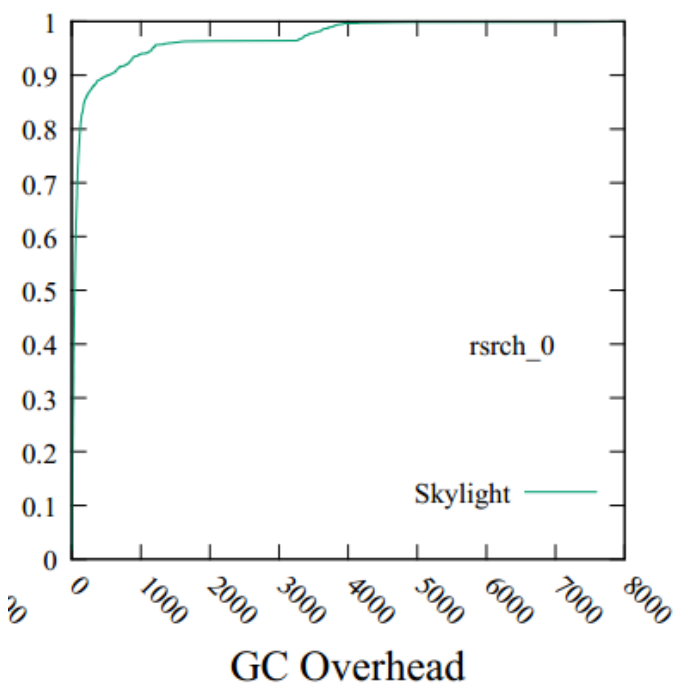


The percentage of amplified writes.

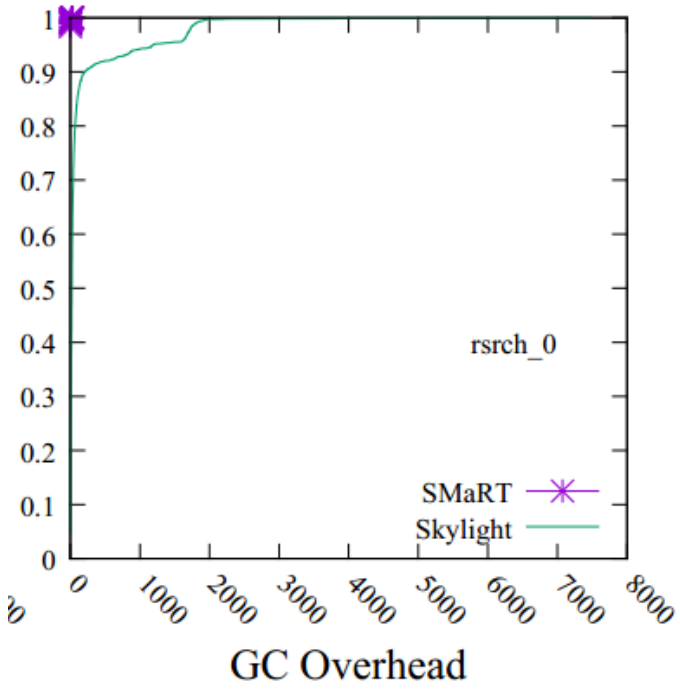
SMaRT has no amplification for 30% but higher amplifications for 60% and 90%.

These numbers are generally low, because both schemes use background GCs.

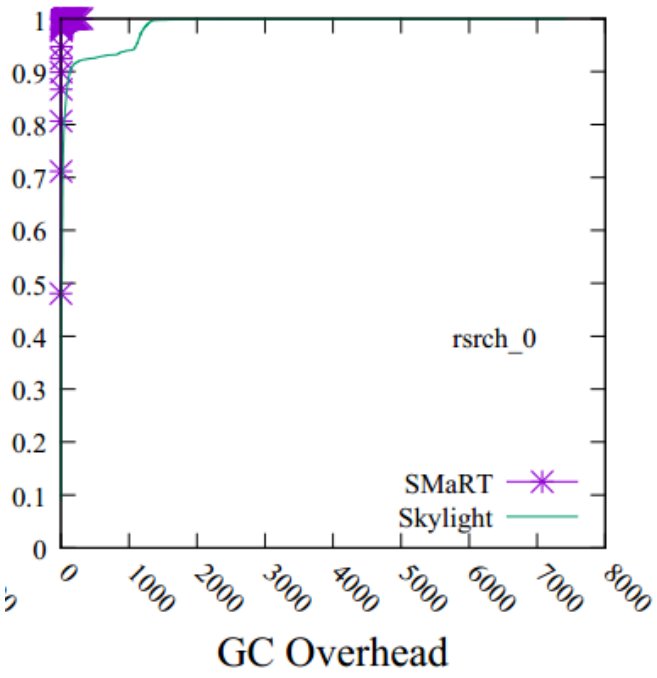
Write Amplification - 2



rsrch_0 @ 30%



rsrch_0@60%



rsrch_0@90%

Write amplification ratio: the number of sub-I/Os created by a single write request.
SMaRT has very narrow spectrum.
Skylight has much wider specturm

Summary

- A DM-SMR solution that exploits inherent SMR properties.
  - Relatively simple and clean
  - No requirement of persistent cache
  - Suitable for primary workloads
  - Friendly to cold write workloads
  - Low metadata overhead

Future Work

- I/O scheduler optimized for SMR drives
- Construct storage system with SMR drives, e.g., RAID and erasure codes
- Hybrid SWDs
- HA-SMR and HM-SMR solutions

References

- [1] http://www.seagate.com/products/enterprise-servers-storage/nearline-storage/archive-hdd/
- [2] http://www.hgst.com/products/hard-drives/ultrastar-archive-ha10
- [3] http:// http://www.open-zfs.org/w/images/2/2a/Host-Aware_SMR-Tim_Feldman.pdf
- [4] A. Aghayev and P. Desnoyers. Skylight—a window on shingled disk operation. In13th USENIX Con-ference on File and Storage Technologies (FAST15)
- [5] Y. Lu, J. Shu, W. Zheng, et al. Extending the life-time of flash-based storage through reducing write amplification from file systems. In FAST, pages 257–270, 2013

www.cris.umn.edu

Thank You!

Questions?