HDD Native Parallel Filesystem

Maximizing Throughput for Highest Capacity CMR & HM-SMR Drives

Piotr Modrzyk

Principal Architect at Leil Storage and SaunaFS

David Gerstein

CTO at Leil Storage and SaunaFS

The International Conference on Massive Storage Systems and Technology
Santa Clara University
September 23, 2025

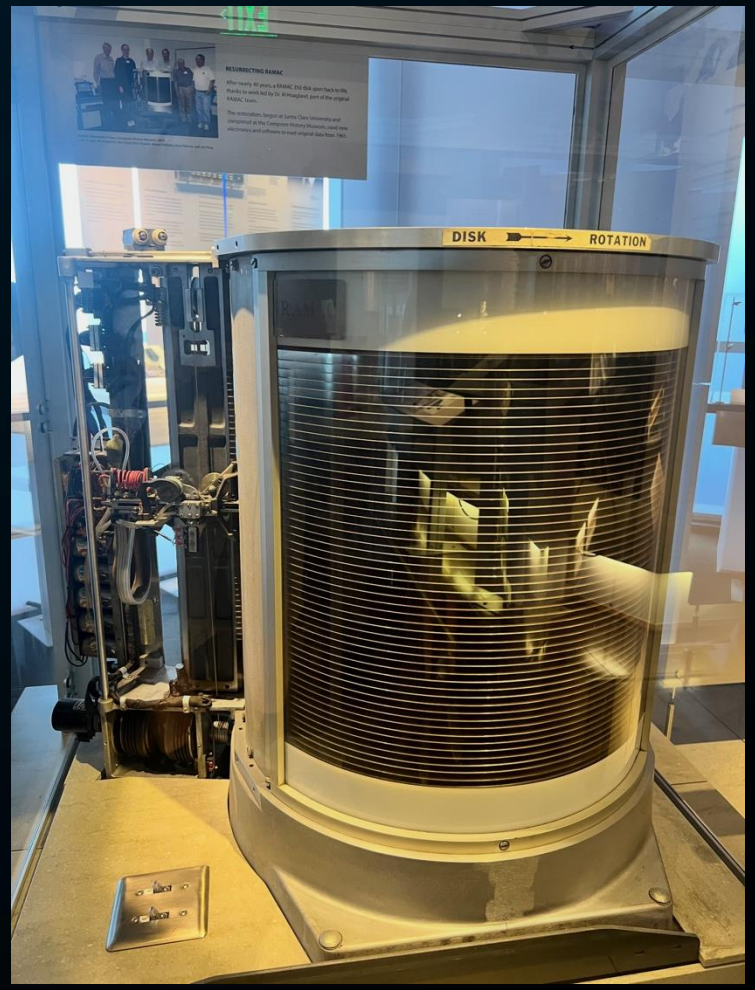
HDDs Through Time: 5MB to 36TB

Shifting Workloads: Random Access to Sequential for HDD Efficiency

- **1956 - IBM 350 RAMAC**: birth of the HDD, designed for random workloads (“R” in RAMAC = Random)
- **Today:** most random I/O has shifted to flash, where IOPS are cheaper than on HDDs
- **HDD strength:** best suited for sequential workloads → our HDD-native approach maximizes throughput by going sequential
- **Longevity:** HDDs remain critical; HM-SMR drives have been widely adopted by hyperscalers for over a decade
- **Next:** let’s dive into HM-SMR and its role today

HDDs Through Time: 5MB to 36TB

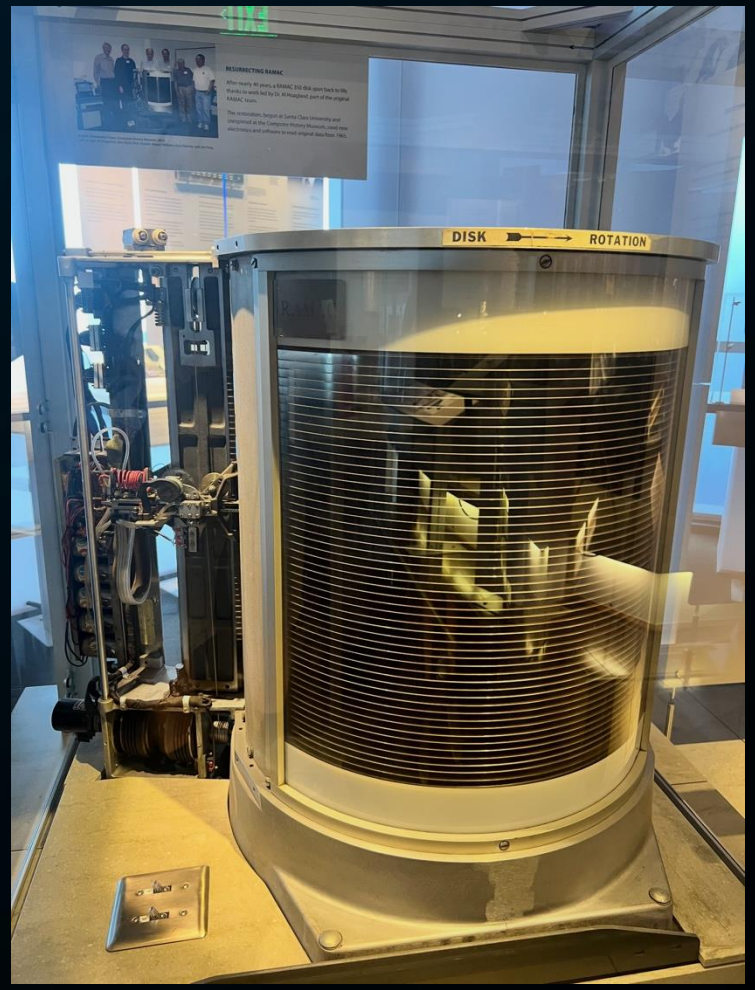
Shifting Workloads: Random Access to Sequential for HDD Efficiency

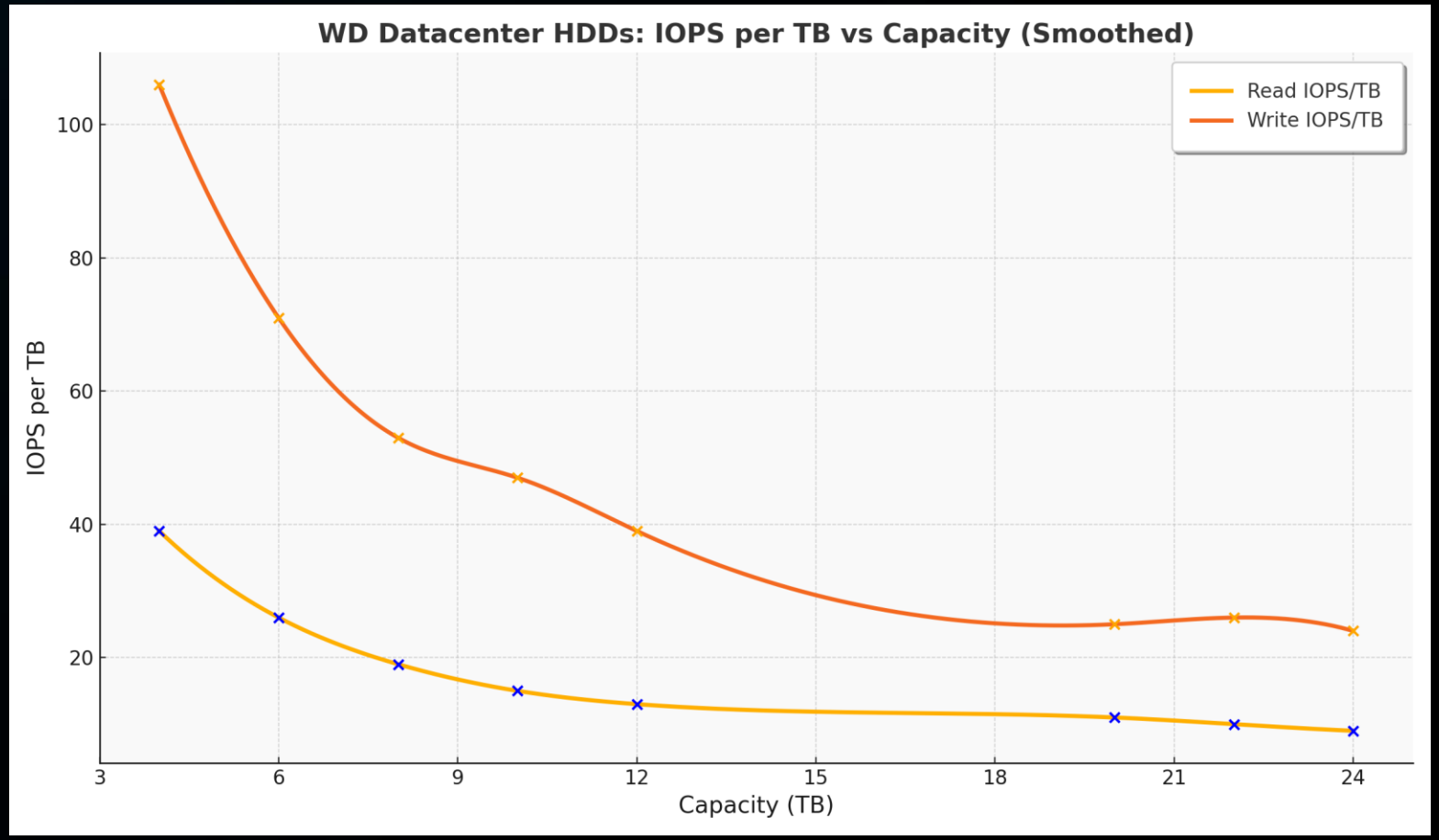


- **1956 - IBM 350 RAMAC**: birth of the HDD, designed for random workloads ("R" in RAMAC = Random)
- **Today:** most random I/O has shifted to flash, where IOPS are cheaper than on HDDs
- **HDD strength:** best suited for sequential workloads → our HDD-native approach maximizes throughput by going sequential
- **Longevity:** HDDs remain critical; HM-SMR drives have been widely adopted by hyperscalers for over a decade
- **Next:** let's dive into HM-SMR and its role today

HDDs Through Time: 5MB to 36TB

Shifting Workloads: Random Access to Sequential for HDD Efficiency





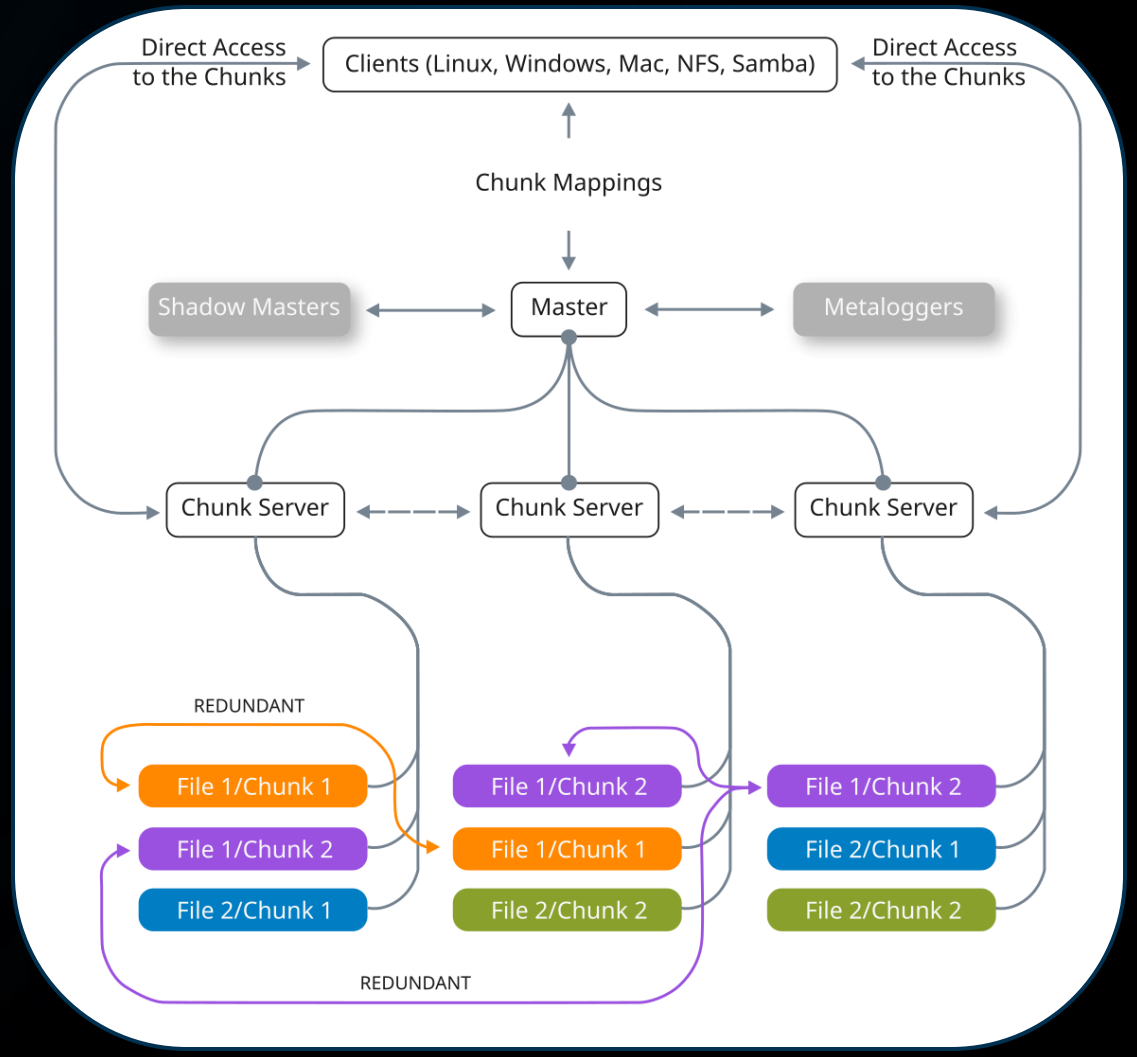
Outline

- Brief intro to SaunaFS
  - Simplified SaunaFS architecture
  - Chunks
- SMR restrictions
  - Problems for conventional Chunks
- Solution
  - Divide the Chunks into Metadata and Data
  - Handle non-sequential writes: fragment the chunks & garbage collection
- SMR libraries overview and why ZoneFS
- Inspecting the content of zones with a graphical UI tool
- Benchmarks & latest improvements
- Hardware Specifics & ADR/RDP plans

SaunaFS - Great-grandson of GFS

leil STORAGE

- Mostly open source & maintained by Leil Storage.
- SaunaFS is a Distributed File System written mostly in C++ which implements concepts introduced by Google File System.
- SaunaFS is divided into:
  - Metadata Servers (master, shadows and metaloggers)
  - Data Servers (chunkservers)
  - Clients (native Linux/Windows, NFS)
- In the Chunkserver side:
  - Files are divided into Chunks (up to 64 MiB)
  - Chunks are logically divided into Blocks of 64 KiB, which is the minimum block size.
  - For each block, 4 bytes of CRC are also stored in the Chunk metadata.



Host-Managed SMR Restrictions

The client wants to create a file and to write data to this file:

- The Sequential Zones can only be appended at the write head – you can NOT write randomly.
- The IO operations must be aligned to the device IO block size (usually 4KiB).

Conventional Chunk

Header 1 KiB

Id, version, type.

CRC

Up to 512 Blocks of 4B.

Data

Up to 1024 Blocks of 64 KiB.

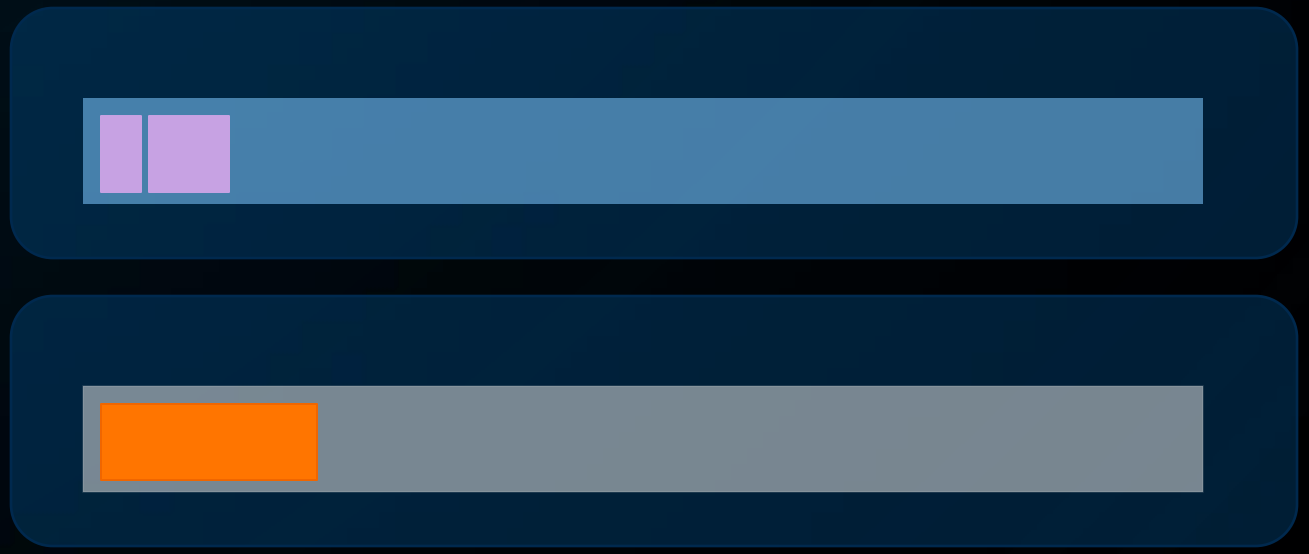
Conventional Chunk Problems

- CRC must be updated with each Block write, which implies non-sequential writes to the Zone.
- Header + CRC = 3 KiB, which is not aligned to the 4 KiB IO block size of many SMRs.
- The Zone write head is always moved by 64 KiB, which only works for write block sizes multiple of 64 KiB.

Solution

Divide Chunks into Metadata & Data

Split the Chunks into Metadata and Data.



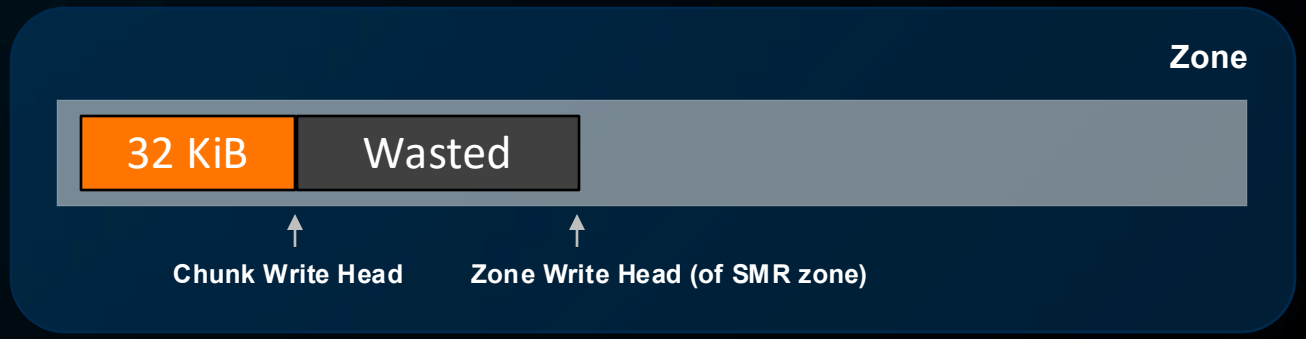
- The metadata is now in another (NVMe) disk (not a cache), which eliminates the problem of writing the CRC in Sequential Zones.
- Data can be aligned now into the Zones with 64 KiB Blocks.
- The Zone Write Head is always moved by 64 KiB, which only works for write block sizes multiple of 64 KiB.

Not Aligned Random Writes

Introduce Chunk Fragmentation

Example of non-sequential write into the Zones:

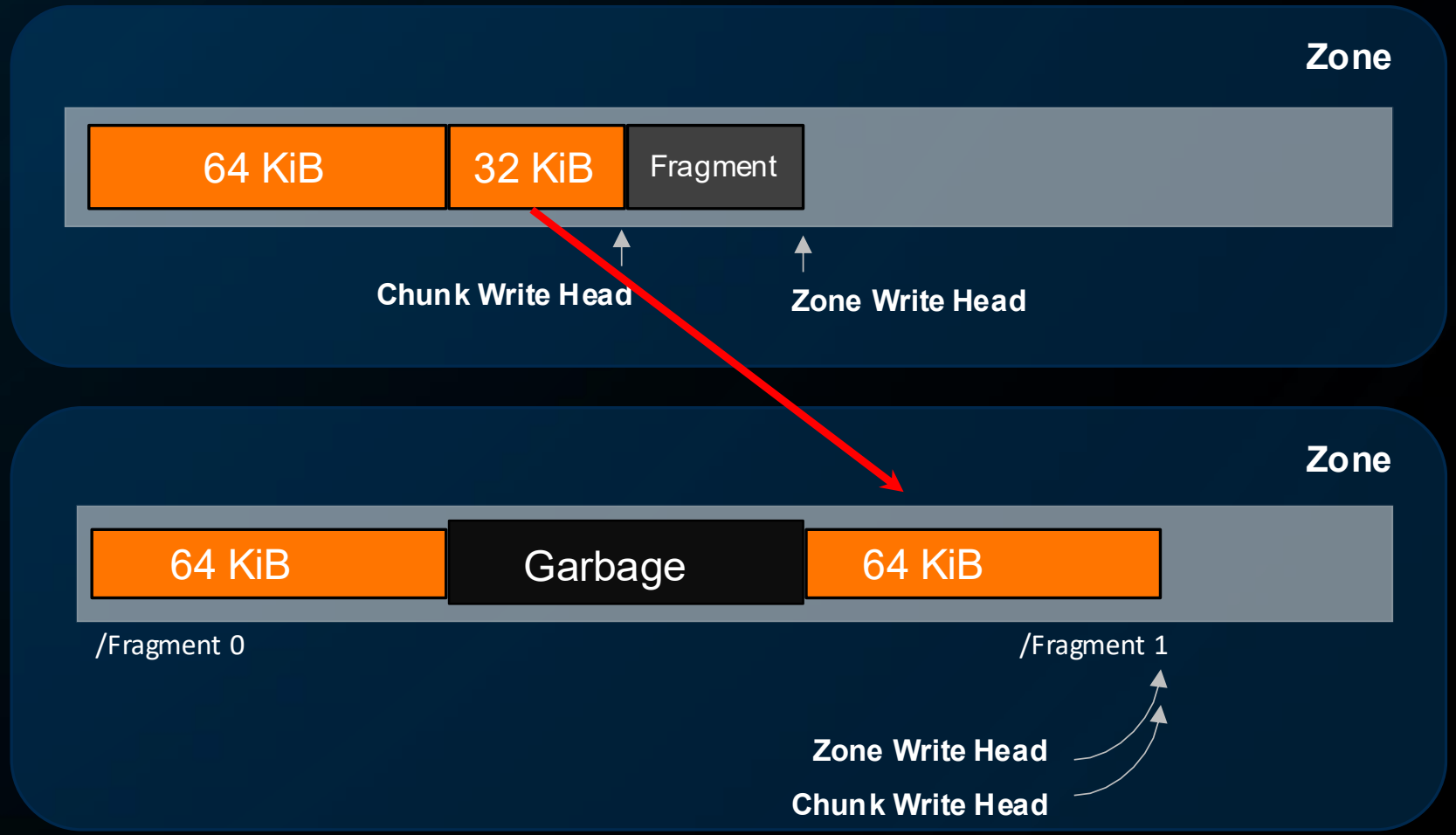
- Create a file and write 32 KiB (50% of our block size).
- A new Chunk is created with 1 Fragment containing 1 Block of 64 KiB, but only 32 KiB belongs to the file.
- The next bytes to write will trigger a non-sequential write into the Zone.



- The next bytes to write will trigger a non-sequential write into the Zone.

Not Aligned Random Writes

Aligning Random Writes



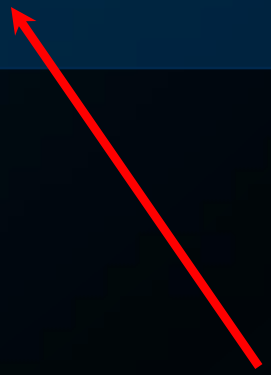
- If the Fragment contains more than one block, we need to create a new Fragment, preferably in the same Zone.
- A hole of unreferenced written data is created.
- The Zone is marked as Dirty.
- The Chunk is now considered fragmented.

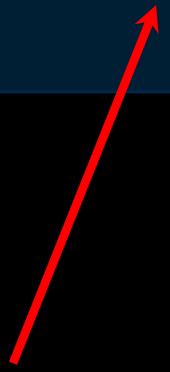
Aligned Random Writes

Handle Non-Sequential Writes

Random Write

```
fio --name=fiotest_rand_write_QD5 --directory=/mnt/saunafs --size=1G
--rw=randwrite --numjobs=1 --ioengine=libaio --group_reporting --bs=64K
```

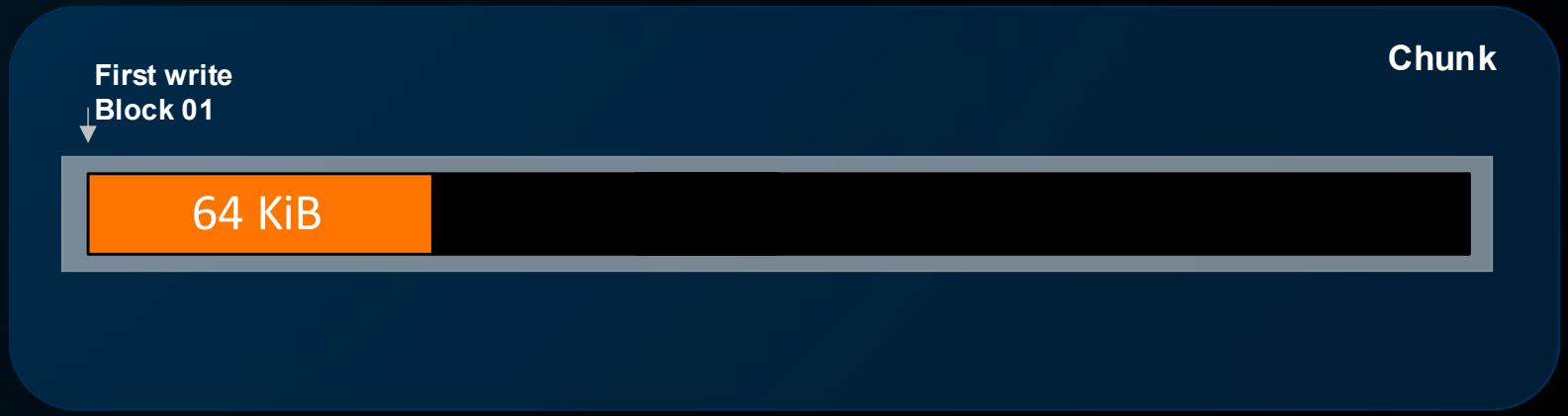


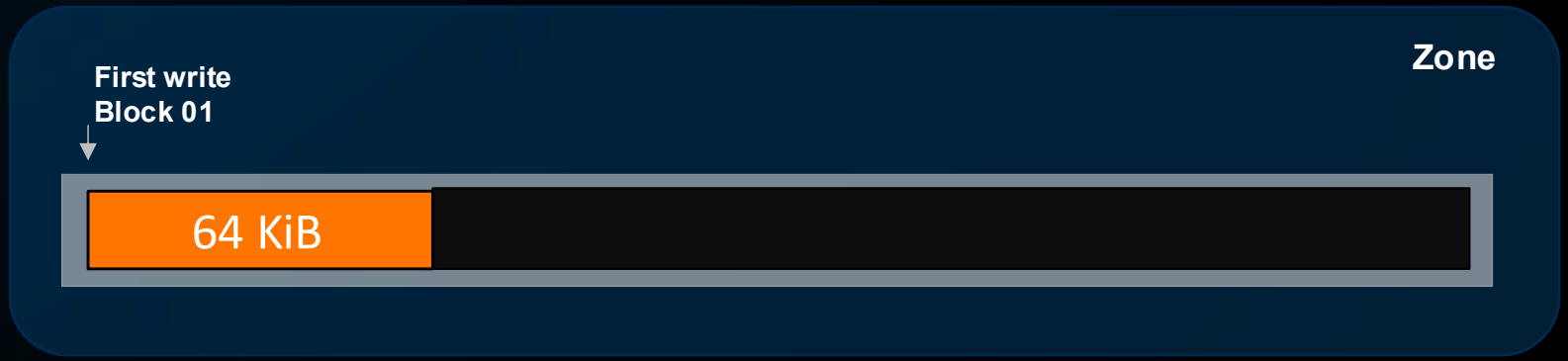


Aligned Random Writes

Handle Non-Sequential Writes

Random Write:

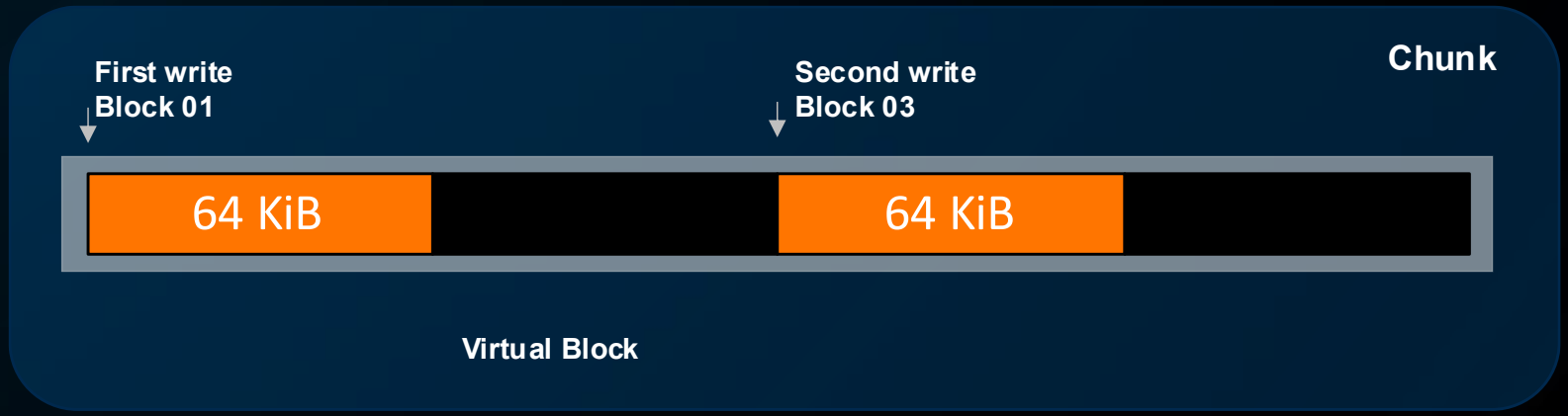


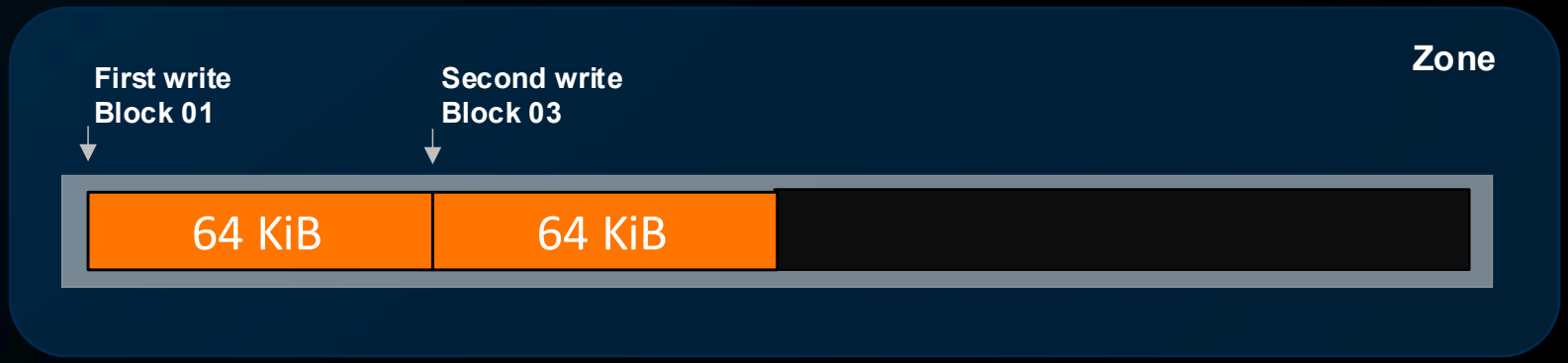


Aligned Random Writes

Handle Non-Sequential Writes

Random Write:





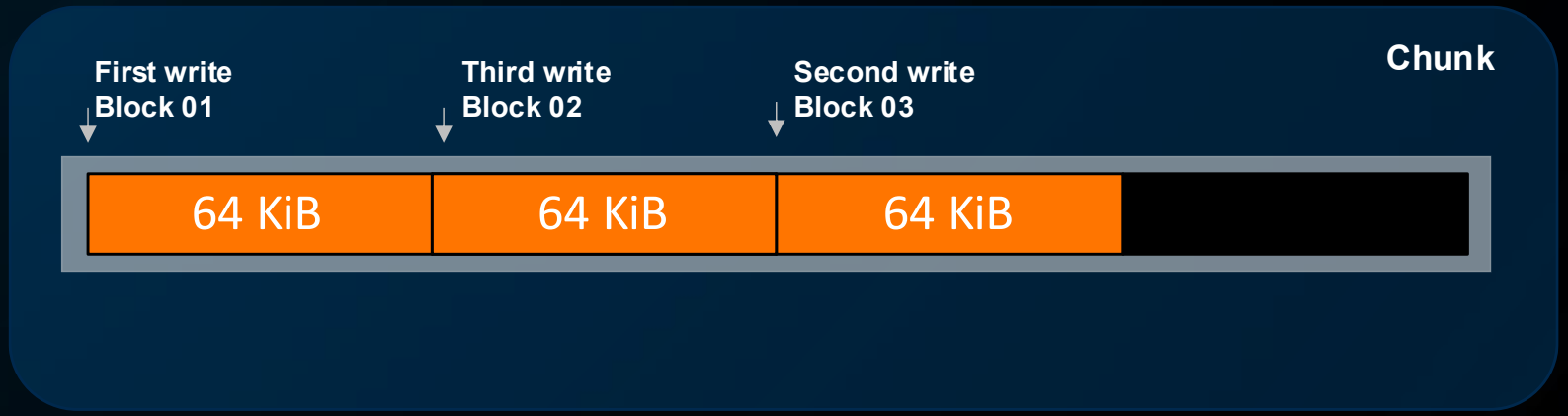
Notice the incorrect order of the blocks in the zone.

The order will be fixed during defragmentation.

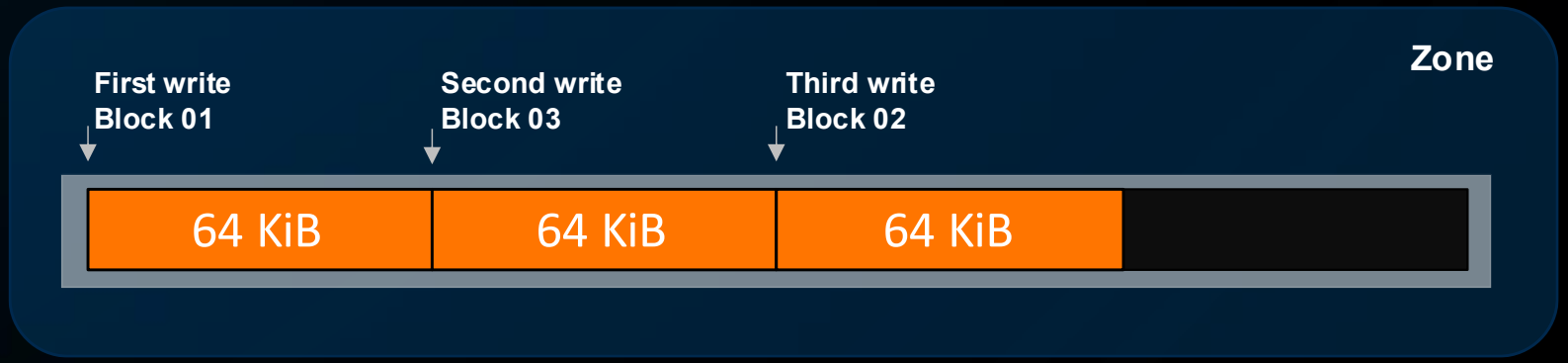
Virtual Blocks

Handle Non-Sequential Writes

Random Write:



Virtual Blocks



Notice the incorrect order of the blocks in the zone.

The order will be fixed during defragmentation.

Virtual Blocks

Introduce Chunk Fragmentation = Virtual Blocks

- The Header is modified to contain information about the Fragments:
  - Id, version, type, number of fragments, list of fragments.
- Metadata about the Fragments contain 12Bytes each:
  - Zone(4B), offsetInZone(4B), first block(2B), number of Blocks(2B).
- New Fragments of same chunk are preferred to be stored in the same Zone if possible.
- A Chunk with more than one Fragment is considered fragmented.

Since we have Virtual Blocks now, defragmentation should be fragment-based instead of the block-based.

This way, we can avoid creating unnecessary Blocks full of zeros each time we would need to deal with virtual block (full of nulls).

Garbage Collection

Chunks Defragmentation in chunk-scrub thread

Defragment the Chunks by extending our scrub-chunk thread, with defragmentation task.

GC is divided into:

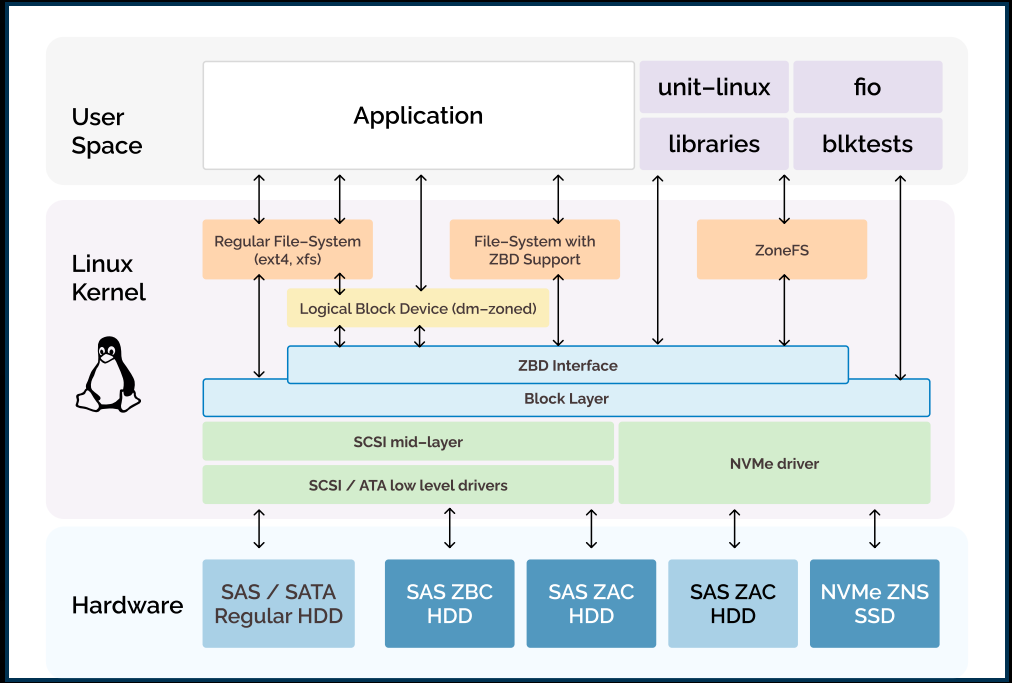
- Defragmenting Chunks.
- Resetting unreferenced Dirty Zones.

The Zone X can be reset now – which will reclaim space, because it does not contain any valid data.

Single HM-SMR Drive Tools

We Are Standing On The Shoulders Of Giants

libzbc	libzbd	ZoneFS
Provides functions for manipulating ZBC and ZAC disks directly.	Provides functions for manipulating zoned block devices (uses the kernel-provided ZBD interface that is based on the ioctl() system calls).	Exposes the zones as files (from kernel 5.6.0). Uses mkzonefs to format the drive and then mount -t zonefs. Provides aggregation for conventional zones, file ownership and file access permissions.
Contains an emulation mode to mimic HM zoned devices.	**No** (but **null_blk** can be used).	**No** (but **null_blk** can be used).
Graphical Interface: **gzbc**.	Graphical Interface: **gzbd**.	**No** (gzbc and gzbd works).



See more details on here:
https://zonedstorage.io/docs/getting-started

Why ZoneFS?

Built-in In Mainstream Kernels

No New Dependencies For The Project like ZBC or ZBD.

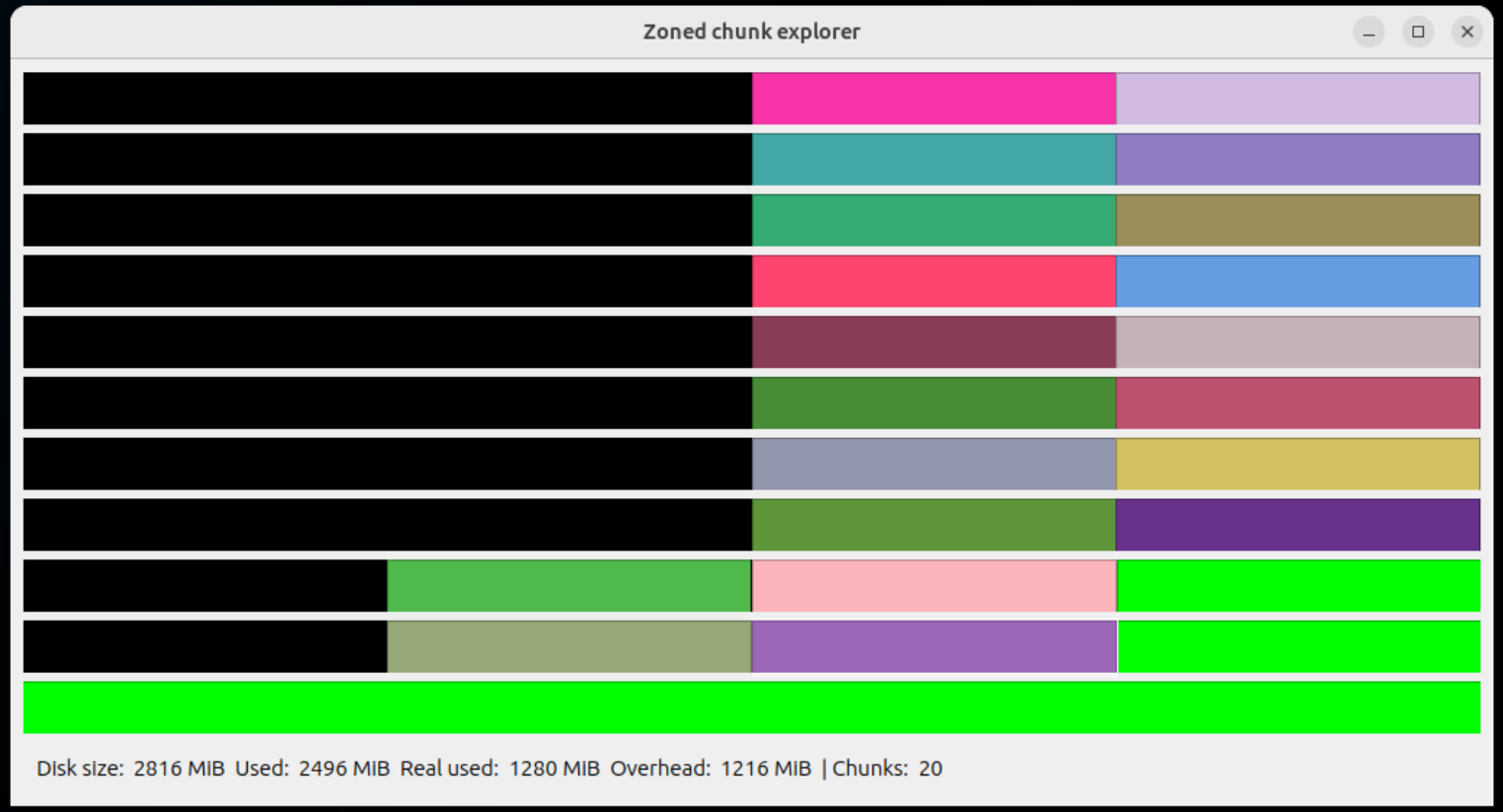
Allows Usage Of Familiar File I/O Model which means less modifications to the current Chunkserver code.

Graphical user interface (GUI)

Garbage collection:

The SaunaFS graphical tool displays each zone as a row.

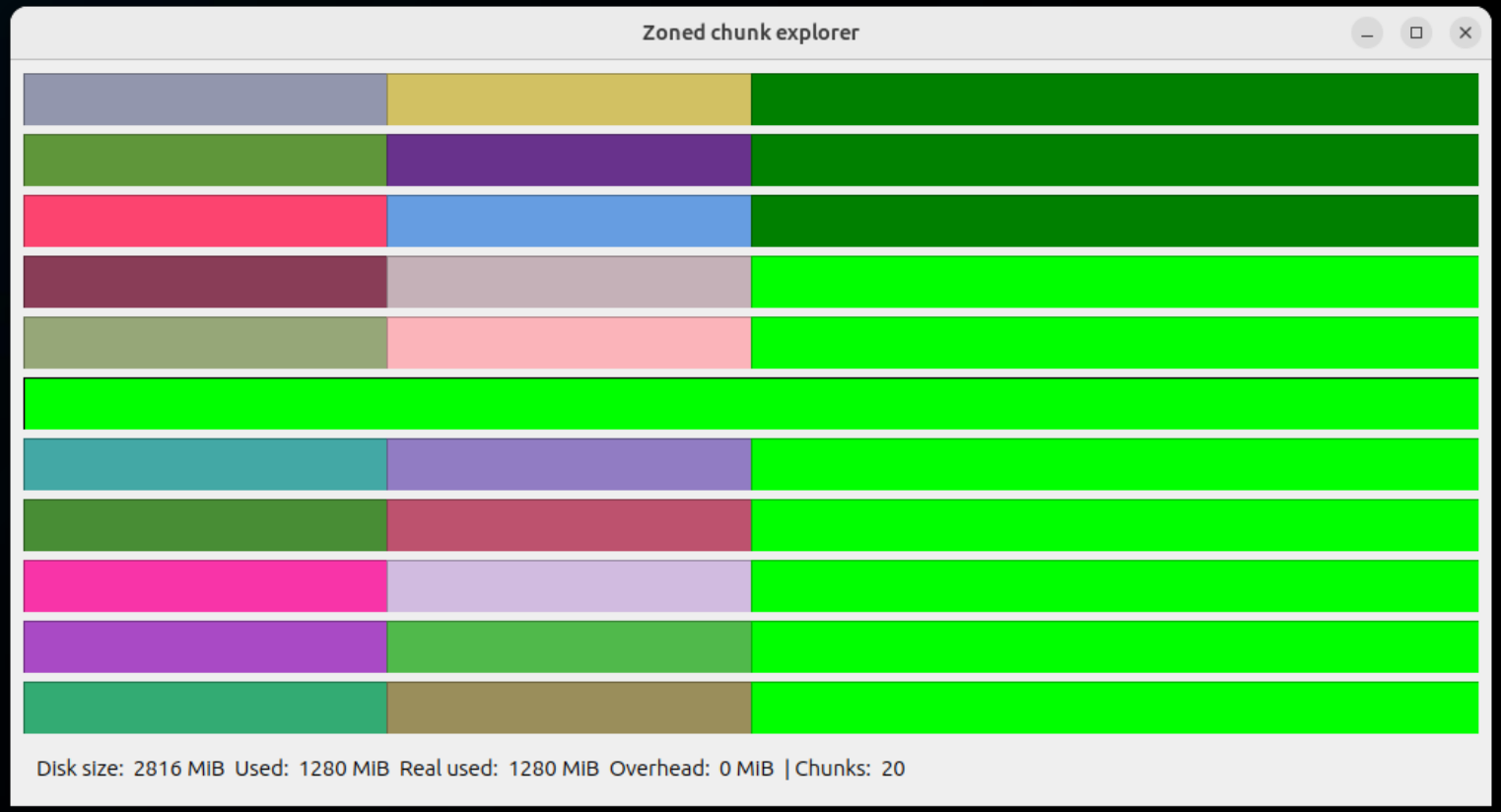
- Black represents holes.
- Each unique color corresponds to a chunk.
- Free space within a zone is shown in **bright green**.



Graphical user interface (GUI)

Garbage collection:

- All chunks in zones are defragmented.
- All dirty (black) space is reclaimed.
- MINIMUM ONE ZONE IS EMPTY.

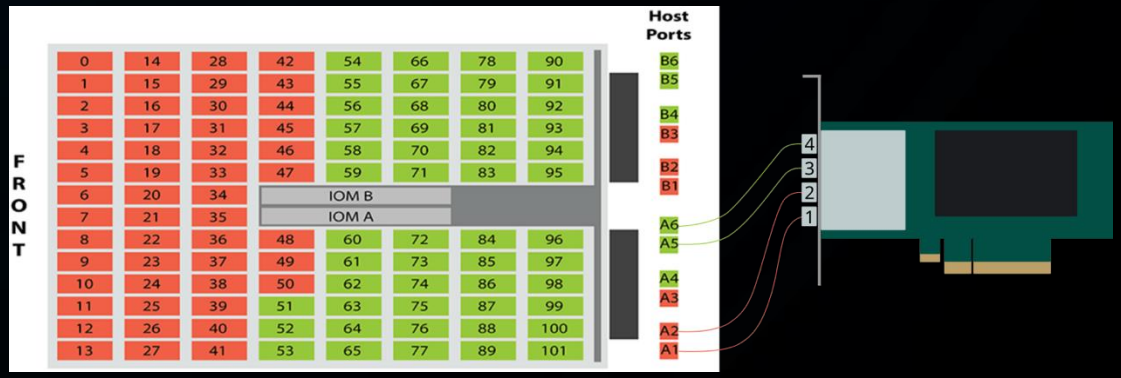


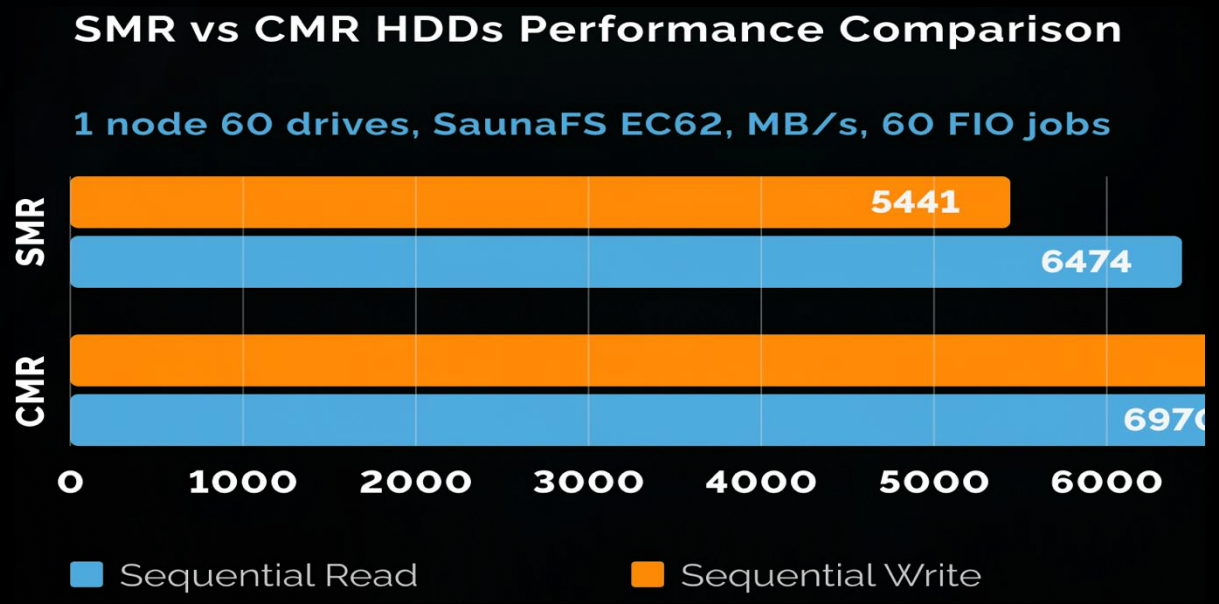
SaunaFS 4.x SMR vs CMR (WDC)

Lab results for the performance comparison

The following graph displays the results for 60 FIO jobs using the following hardware:

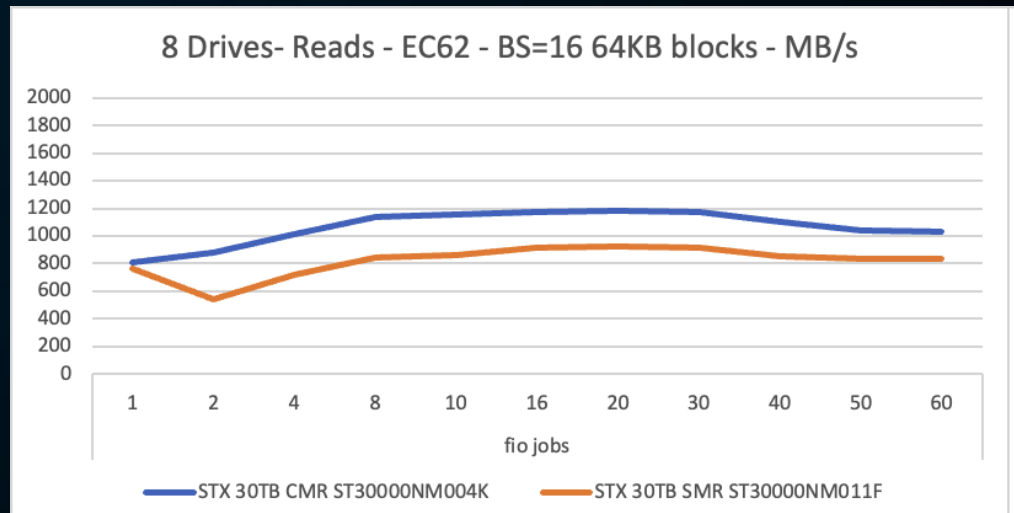
- CMR Drives: Ultrastar HC560 (SAS)
- SMR Drives: Ultrastar HC650 (SATA)
- Platform: Ultrastar Data102 in a single IOM configuration was used for all tests

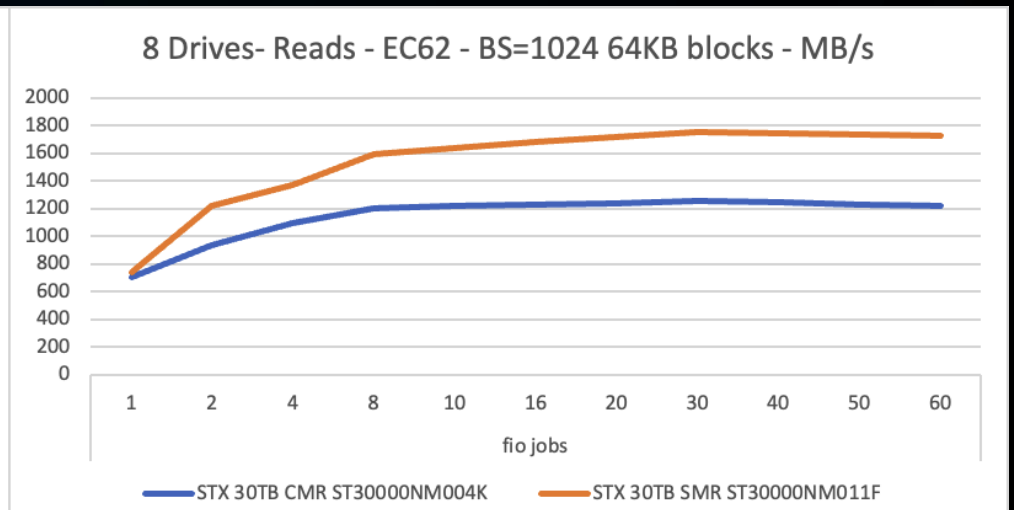


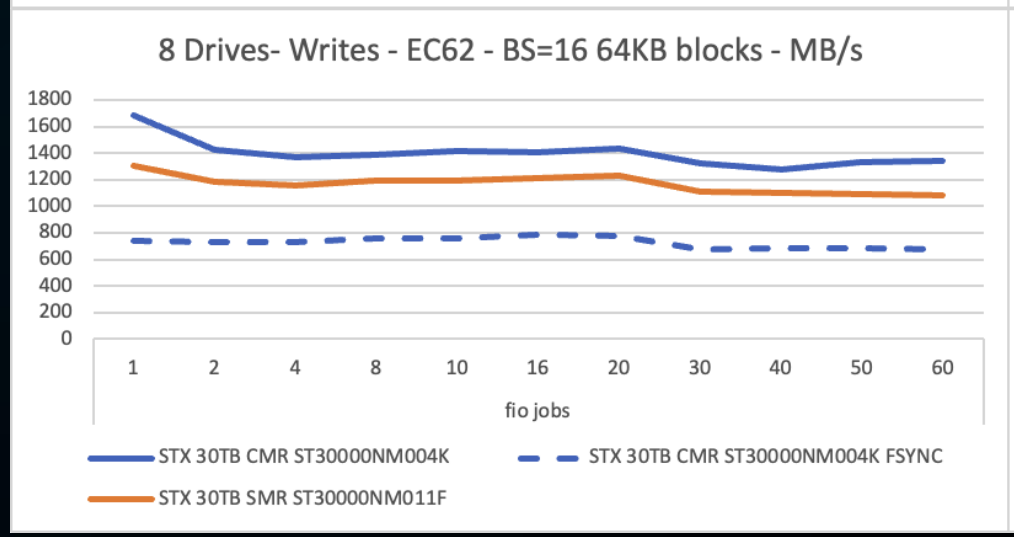


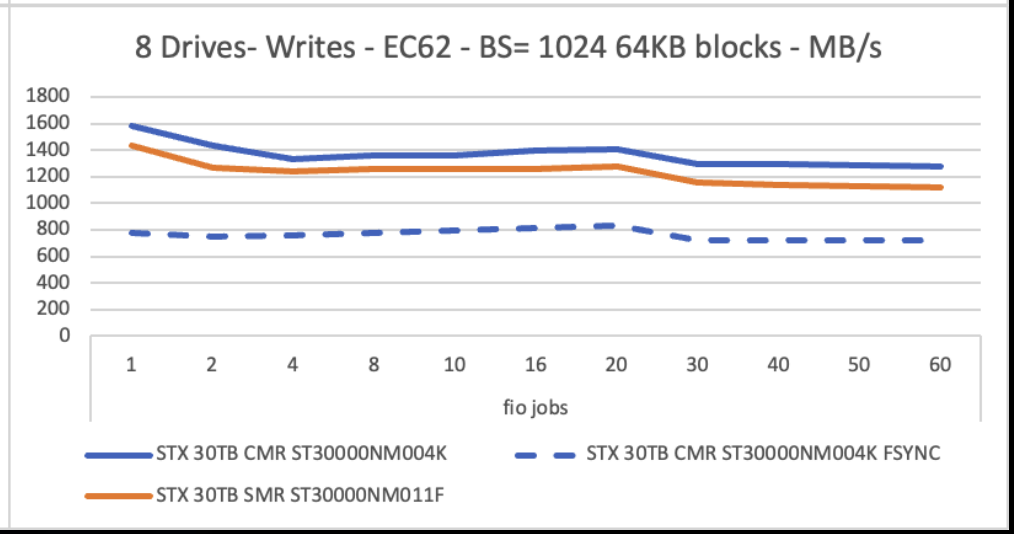
More information available in the Solution Brief:
[Leil Storage Partners with Western Digital to Deliver a Distributed File System Enabling Host-Managed SMR at Petabyte Scale without Cloud]

SaunaFS 5.x SMR vs CMR (STX)





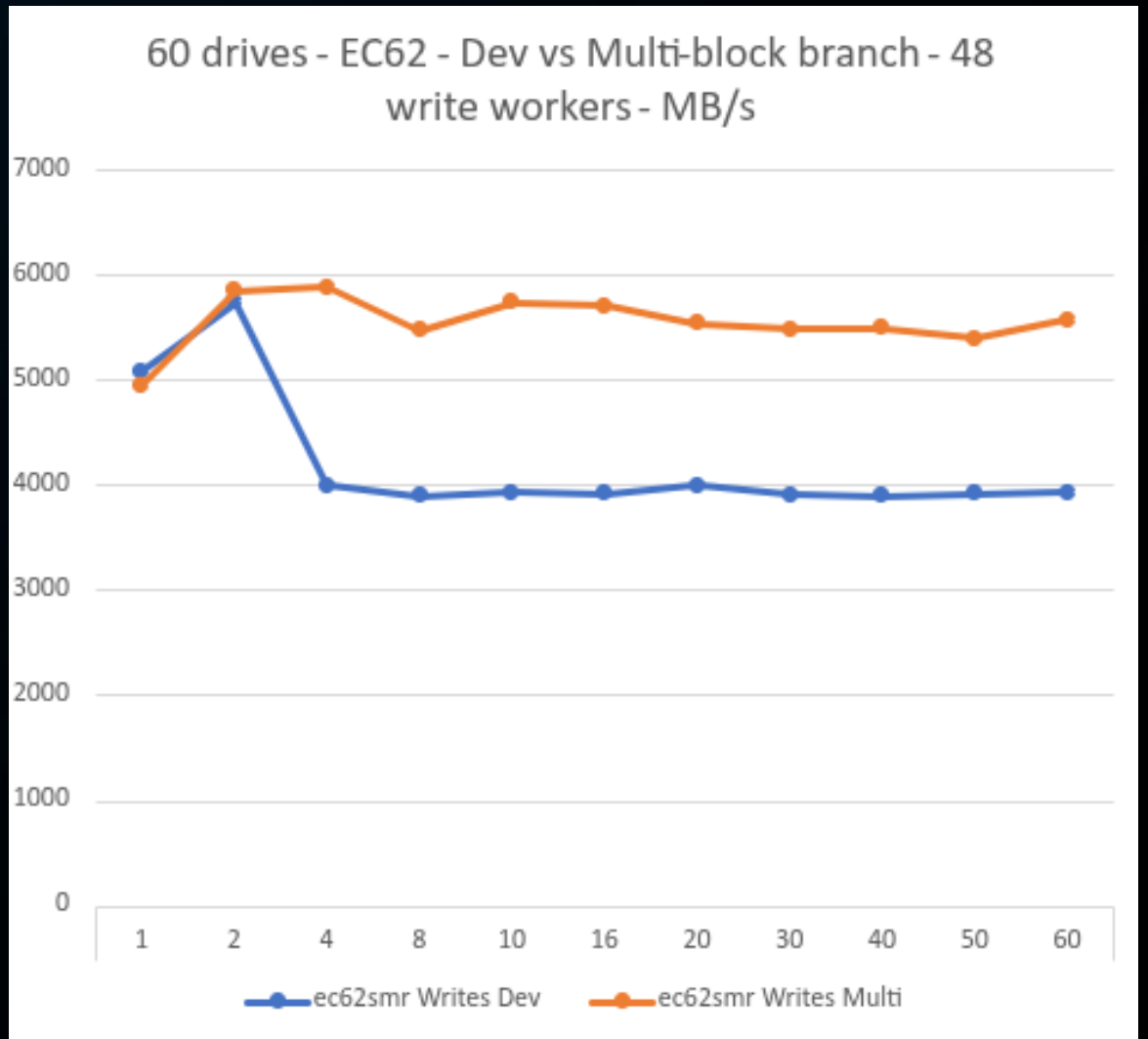




* SMR benchmark performed with ZoneFS - direct I/O; CMR tests with XFS - buffered I/O

SaunaFS 5.x Recent Improvements

Recent improvements



Status as of version 5.1.0:

- The 3rd generation of HM-SMR support is stable with a room for further performance improvements.
- Both Read and Write performance was improved (25-30% depending on the workload) with new write and zone assignment and read-scheduling algorithms.
- SMR to CMR write performance gap has been significantly reduced.

Leil Reference Architecture

Leil Storage HM-SMR Hardware Blueprint

Data Scheme: EC6+2, 8 nodes up to 11.5 PB usable in a single rack

Node Architecture:

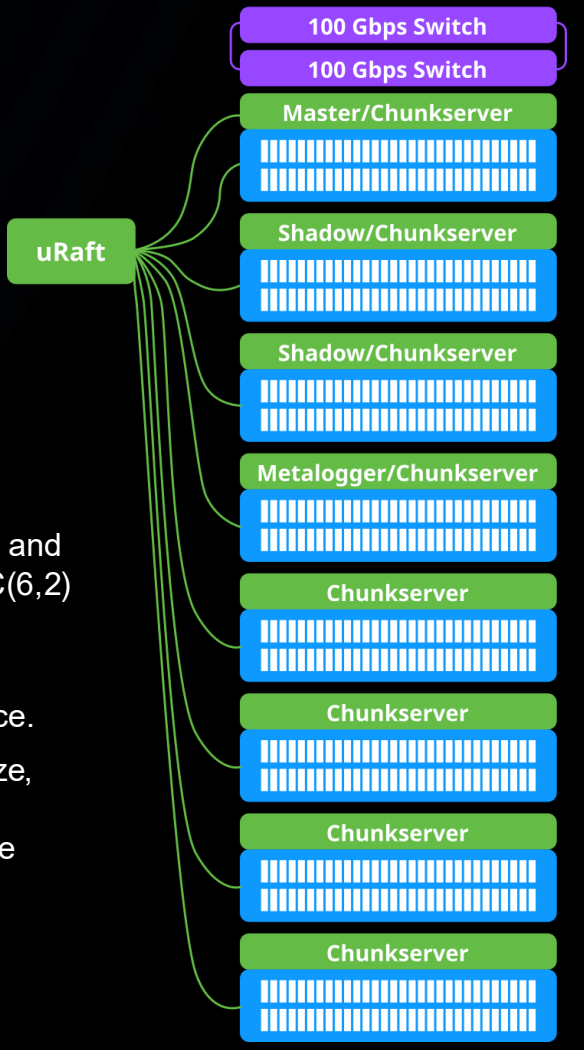
AIC SB407-ZL 4U High-Density Server

- Up to 1.9PB of raw storage in 4U
- 60x 32TB HM-SMR SATA HDD with PIN3
- 2x U.2 7.68TB U.2 NVMe metadata drives
- 1x AMD Siena 8224P (24 cores, 2.55GHz)
- 6x 96GB DDR5 ECC
- 2x 960GB U.2 NVMe OS drives
- 1x Dual-port 100GbE QSFP28 NIC
- 1x ATTO H240F 24Gb HBA

- Best TCO for 60 drives node sizing
- Fits in standard 1m depth rack
- OEM Ready

SaunaFS Architecture:

- Up to 1.9PB of raw storage in 4U
- 8 server nodes, 60 high-capacity drives per node.
- Erasure Coding with 6 data parts and 2 parities, internally known as EC(6,2) or simply EC62.
- Using 32 TB drives, a single rack could have about 11.5 PB of space.
- Depending on the average file size, the number of metadata could represent a bottleneck for a single active Master.

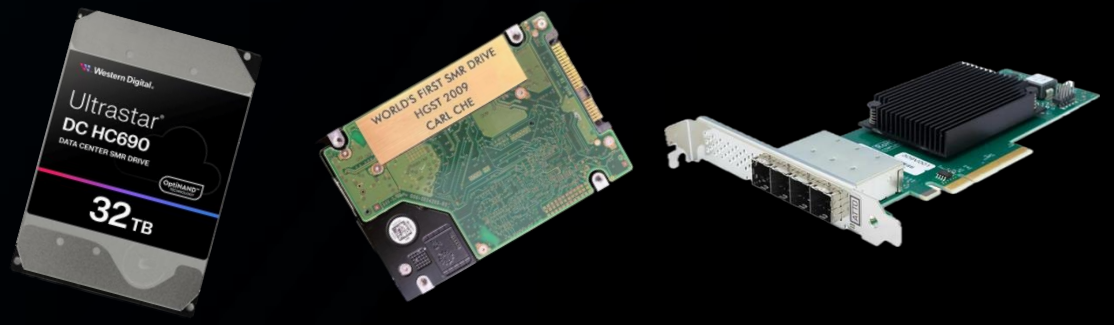


WD Reference Architecture

Boosting Performance with Zoning Setup

- Ports 1-2 (wide port) of ATTO HBA are connected to A1 & A2 (red zone)
- Ports 3-4 (wide port) are connected to A5 & A6.
- We do not use the second IOM because we are using SATA drives (same will be for new gen JBODs 3000 Series with 24Gb HBAs).

HBA
1x ATTO ESAH-12F0-GT0:
16-port 12Gb HBA



HM-SMR Drives
Single-port 6 Gb/s SATA

To Host
2x Mini-SAS HD ports per zone
4x Mini-SAS HD ports total

ADR/RDP for HM-SMR vs CMRs

leil STORAGE

Autonomous Drive Regeneration (ADR) / RDP (Repurposing Depopulation) for HM-SMRs

How it Works

- Depopulates failing head (~5% capacity loss in 10-platter drive).
- With HM-SMR: stop only zones of bad head → **no reformat, no downtime**.
- Data stays in place; reconstruct via **EC** if needed (**damaged zones only**).

Benefits

- Avoids **24h+ full reformat** (data destructive).
- **Zero downtime** and longer drive life.
- 5% surface loss reduces data-loss probability by **20× with EC (N+M) when M drives are lost.**

Next on SaunaFS RoadMap

Thank you

Piotr Modrzyk

Principal Architect at Leil Storage and SaunaFS

David Gerstein

CTO at Leil Storage and SaunaFS