

Lustre on Shingles: Enabling SMR HDDs with XFS

Matias Bjørling, Distinguished Engineer
Emerging System Architectures, CTO Office & Research

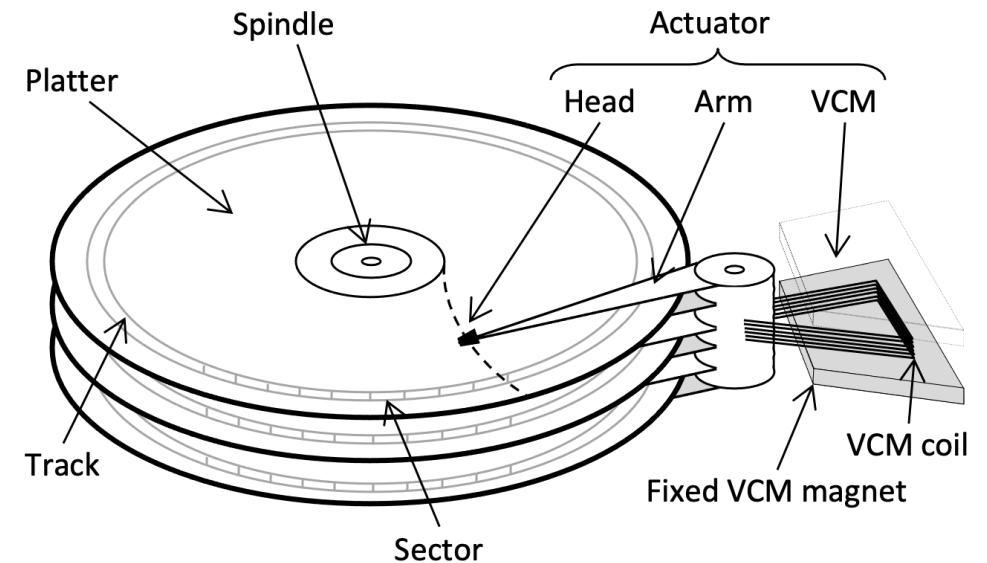
September 2026

Overview

- | | |
|----|-------------------------------|
| 01 | Introduction |
| 02 | Zoned Storage Ecosystem & XFS |
| 03 | Shingling Lustre with XFS |
| 04 | Introducing osd-xfs |
| 05 | Evaluation |
| 06 | New Features and Summary |

HDD 101

- Platters (media) accessed with heads (read and write heads)
 - All heads move together, with only a single head active at any time
- Platters are formatted as a set of concentric tracks of sectors
 - Old CHS addressing: all platters had the same number of tracks (cylinders) with each track having the same number of sectors
 - LBA addressing improved changed this to increase capacity
 - Longer tracks at the outer diameter of the disk have more sectors, and as a result have a higher throughput
- Storage cost (lower \$/TB) is reduced with increased capacities, realized with evolving recording technologies



HDD Magnetic Recording (MR) Technologies

EAMR: Energy Assisted MR

Use higher magnetic coercivity media to increase bit density. Requires Energy assist to temporarily reduce coercivity when writing

Longitudinal
MR

PMR
Perpendicular MR

MAMR
Micro-wave MR

ePMR

HAMR
Heat Assisted MR

RAMAC
1954

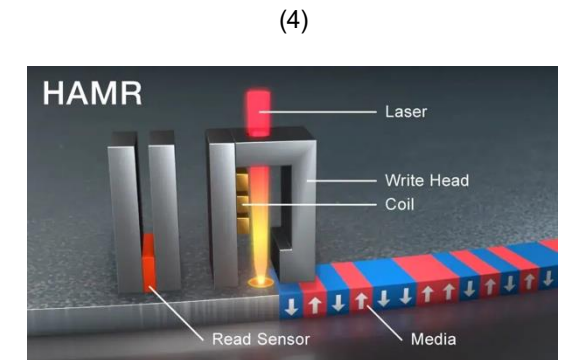
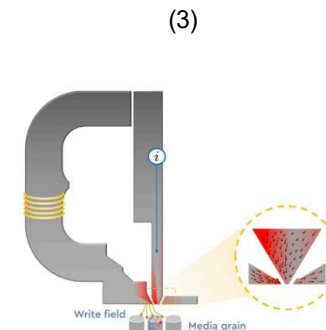
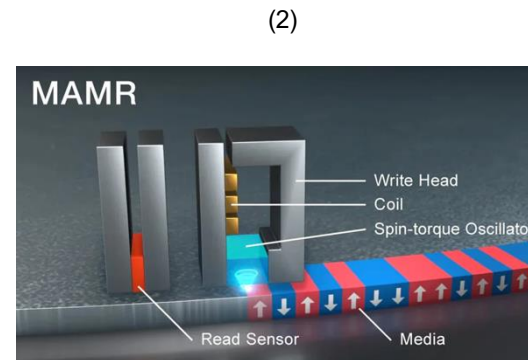
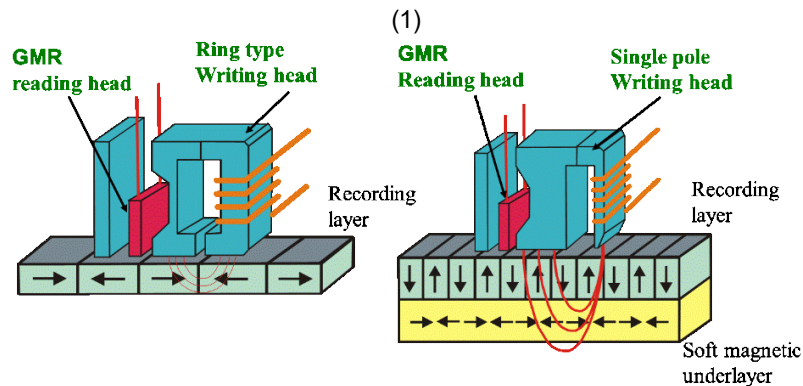
1980~1994
CHS -> LBA
addressing

~2010

2021

2024

2025



(1) https://ethw.org/Milestones:Perpendicular_Magnetic_Recording,_1977

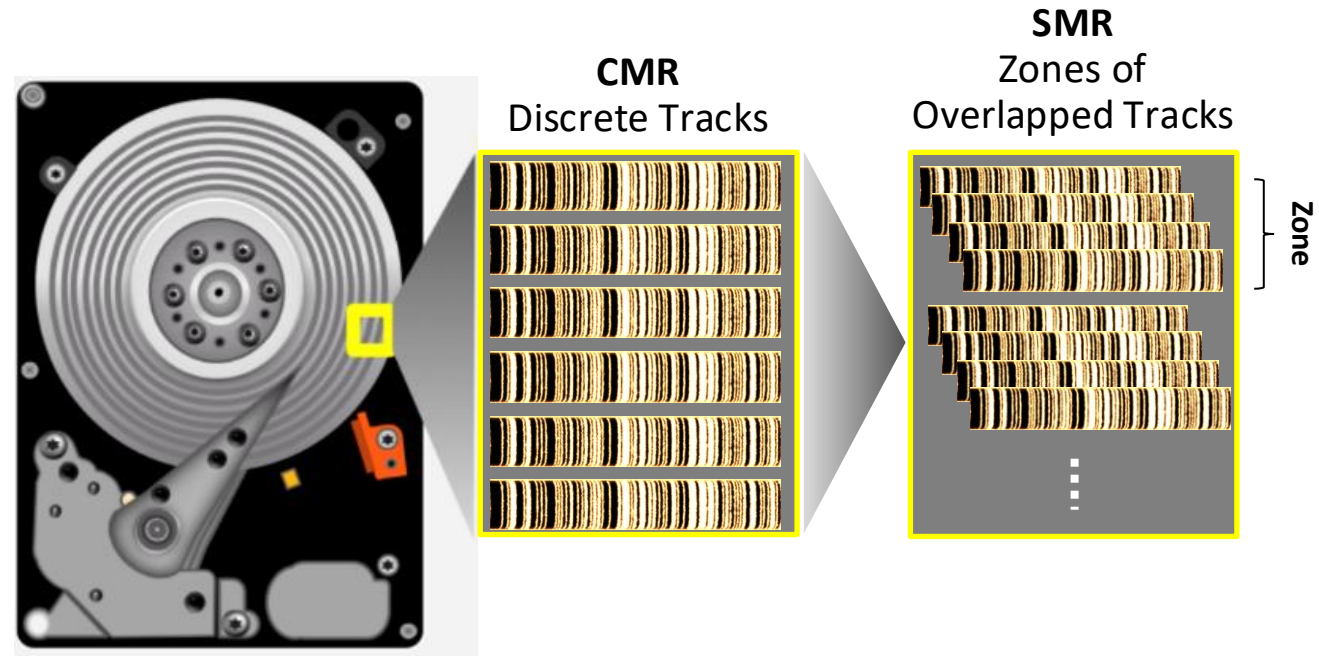
(2, 4) https://www.tdk.com/en/featured_stories/entry_025.html

(3) <https://www.tomshardware.com/news/epmr-emar-new-tech--wd-gold-16tb-18tb-hdd>

© 2026 Western Digital Corporation. All rights reserved.

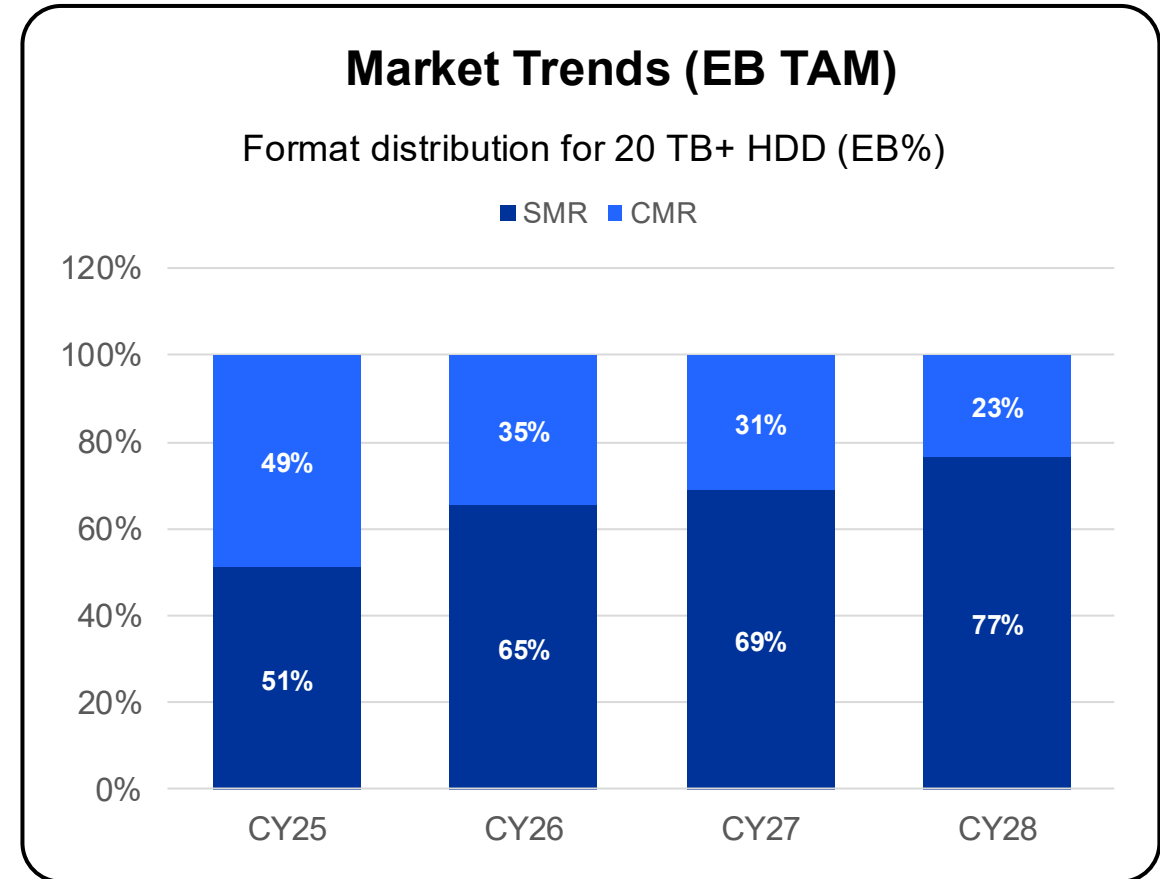
Conventional vs Shingled Magnetic Recording

- Conventional Magnetic Recording (CMR): non-overlapping discrete tracks
 - Sectors can be read and written randomly
 - Shingled Magnetic Recording (SMR): zones of overlapping tracks
 - Sectors can be read randomly but must be written sequentially in each zone
-
- 20 to 25% capacity increase with more efficient use of media space
 - Sequential write pattern enables stronger ECC/data encoding for better reliability
 - Applies to all recording technology (e.g. SMR on HAMR)



SMR is now Mainstream

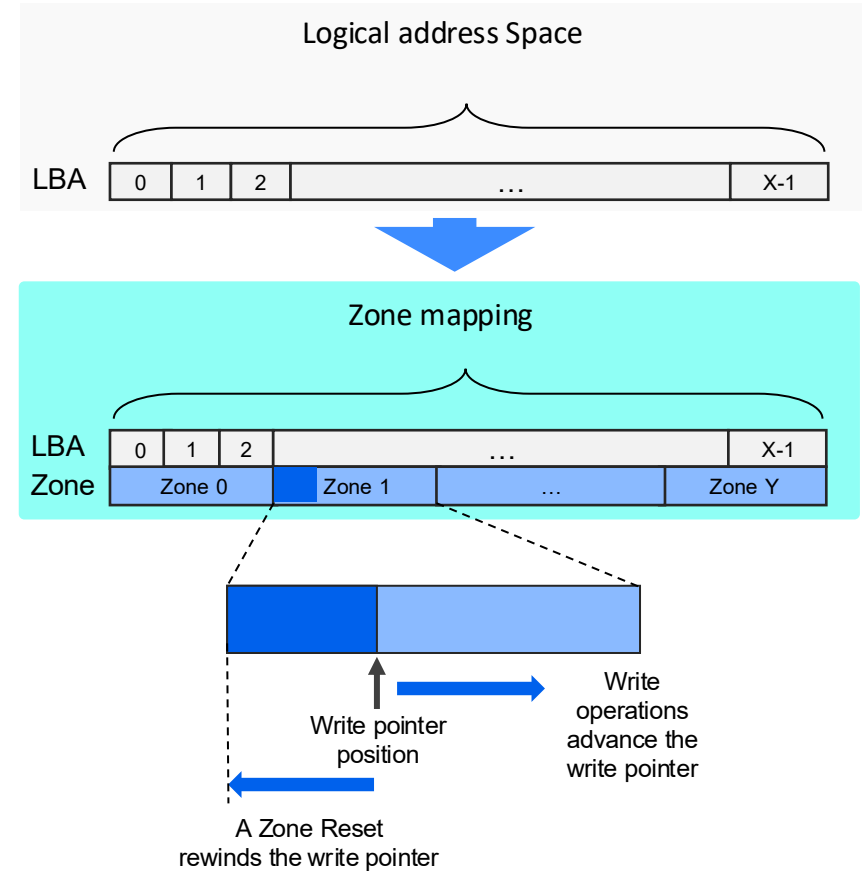
- Significant value to Cloud and AI workloads by enabling more storage capacity and reliability
 - Adoption has accelerated and over 65% of the Exabytes shipped in 2026 will be SMR
- CMR is increasingly used only in “legacy” applications
 - Small scale storage systems



20TB+ estimated 88% of WDC EB in CY25 and 93% in CY27.
Sources: Estimated based on IDC Hard Disk Drive Forecast May 2026.

The Zoned Storage Programming Model

- **Zoned storage devices** (i.e., host-manager SMR HDDs) expose capacity as a set of **zones** rather than a flat random-write address space.
 - Each zone must be written **sequentially** from its current **write pointer**.
 - A zone typically transitions through states such as: **Empty** → **Open** → **Closed/Full** → **Reset**
 - To reuse space, software must **reset** a zone before writing from the beginning again.
- **Core rules**
 - **Reads:** allowed from any written area in a zone
 - **Writes:** must be **sequential append-style**
 - **Overwrite in place:** not allowed
 - **Space reclamation:** done by **moving valid data** elsewhere, then **resetting** the zone



The Zoned Storage Linux® Ecosystem

In development since 2016

- Zoned Kernel API since kernel version 4.10 (Feb 2017)

5+ Linux Distributions with Zoned Storage Support

- RHEL 9+, CentOS 7+, Fedora 33+, Debian 11+, and Ubuntu 21.04+

Local File-systems

- f2fs (mobile/embedded - UFS) and btrfs
- XFS - no longer experimental from 7.2 onwards

Storage Systems

- Ceph, OpenEBS, Mayastor, SPDK's CSAL, ...

Library/Tools support

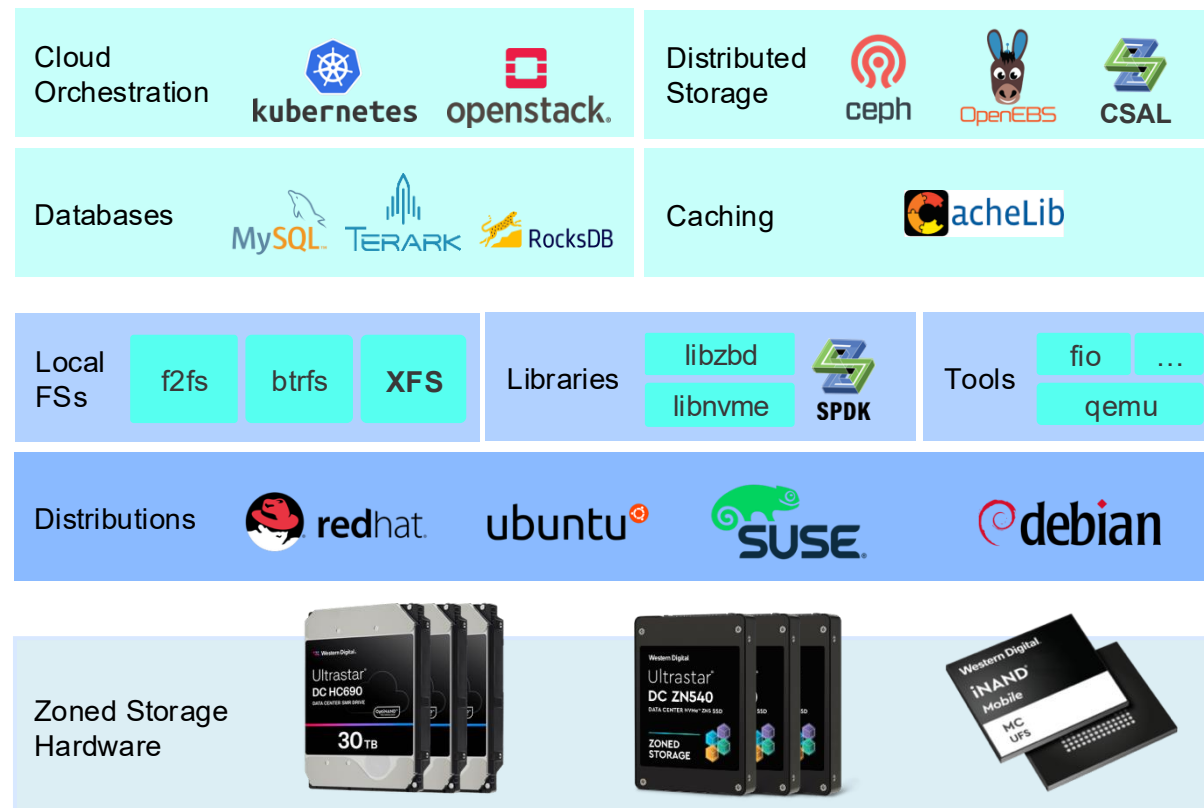
- libzbd, libnvme, SPDK, fio, qemu, blkzone, blktests, ...

End-to-end Applications

- Cloud Orchestration Platforms, Databases, Caching, and Distributed Storage

Mature, robust, and used in production by some of the biggest consumers of storage

© 2020 Western Digital Corporation. All rights reserved.



Shingling Lustre

- How to enable Lustre to support SMR HDDs for the future?
- Lustre has two main storage backends:
 - Idiskfs – ext4 with patches on top. Maintained out of tree. The main storage system for lustre
 - ZFS – OpenZFS – maintained out of tree
- Both storage backend's require in-place updates and currently not compatible with the zoned storage writing requirements. Most likely require on-disk format changes to support.
- Any other file-system candidates? How about XFS?



*AI Generated. THIS IS NOT HOW AN HDD LOOKS ;)

The XFS File-system

- Proven enterprise filesystem with over 30 years of development
 - Designed for large-scale storage and highly parallel I/O
 - Scalable metadata architecture minimizes contention
 - Allocation Groups provide a natural mapping to zoned media
- Any other file-systems that supports SMR? Yes;
 - btrfs – Possible, but its zoned storage support is not as mature as XFS'
 - f2fs – The file-system targets mobile and embedded devices and only supports up to 16TB storage devices. Not applicable to Lustre's workloads.

The logo for the XFS file system, featuring the letters 'XFS' in a bold, dark blue, sans-serif font. A small registered trademark symbol (®) is located at the top right of the 'S'.

Bringing Zoned Storage to XFS

Work started late 2024, led by Christoph Hellwig.

Introduced experimental support into Linux kernel 6.15 (May 2025) and marked stable from Linux kernel 7.2 (August 2026) onwards.

- **Separate Data from Metadata and Journaling**
 - Isolate user data writes from metadata and log traffic to accommodate different update patterns.
 - Preserve existing on-disk formats and filesystem compatibility.
- **Decouple Physical and Logical Space Accounting**
 - Manage physical zone occupancy independently from user-visible free space.
 - Account for zone states, write pointers, and relocation overheads.

- **Develop Zone-Aware Allocation Policies**
 - Minimize fragmentation and maintain sequential write efficiency.
 - Employ placement heuristics that keep zones homogeneous and easier to reclaim. Avoid unnecessary data movement whenever possible.
- **Introduce Online Garbage Collection**
 - Reclaim space from partially utilized zones while preserving filesystem consistency.
 - Operate transparently to applications and workloads.
 - Provide robust mechanisms for migrating live data during zone reclamation.
 - Minimize performance impact while maintaining crash consistency.

Introducing osd-xfs

- **A PoC storage backend to understand the feature requirements and feasibility of what it would take to support XFS & SMR HDDs in Lustre**
- **Design goals**
 1. Focus on primarily OST-based workloads. I.e., continue to use Idiskfs for MDTs.
 2. Scope features between Idiskfs and XFS
 - Quota, transactions, naming scheme, ...
 3. Scope necessary changes to the Linux kernel. Fewer changes, the better.
 4. Achieve ballpark performance parity with Idiskfs



Implementation

Detail	What?	ldiskfs	xfs	Comment
File ID Attribute	Allows fast name to FID lookup	Stored as part of dentry	Stored as an extended inode attribute (same inode. I.e., single inode read as ldiskfs)	FID = {Seq. #, Obj #, Version)
Parallel Directory Operations	Allow many threads to insert simultaneously into	Htree implementation	Assumes osd-xfs only as OST (i.e., client does not see the same dir. structure). Implements directory buckets to spread lock overhead.	The concurrency comes from the object layout: O/<sequence>/d0/<object> O/<sequence>/d1/<object> ... O/<sequence>/d31/<object>
Multi-object Operations	Ability to commit multiple objects in one go	Supported through the on-top patches.	Not yet implemented an equivalent in XFS	Impacts overall write performance.
FIEMAP kernel access	Access fiemap from kernel users	Kernel patch	Kernel patch	FIEMAP asks a filesystem which parts of a file are allocated, unwritten or holes. E.g., Clients filefrag a file and layout is needed.

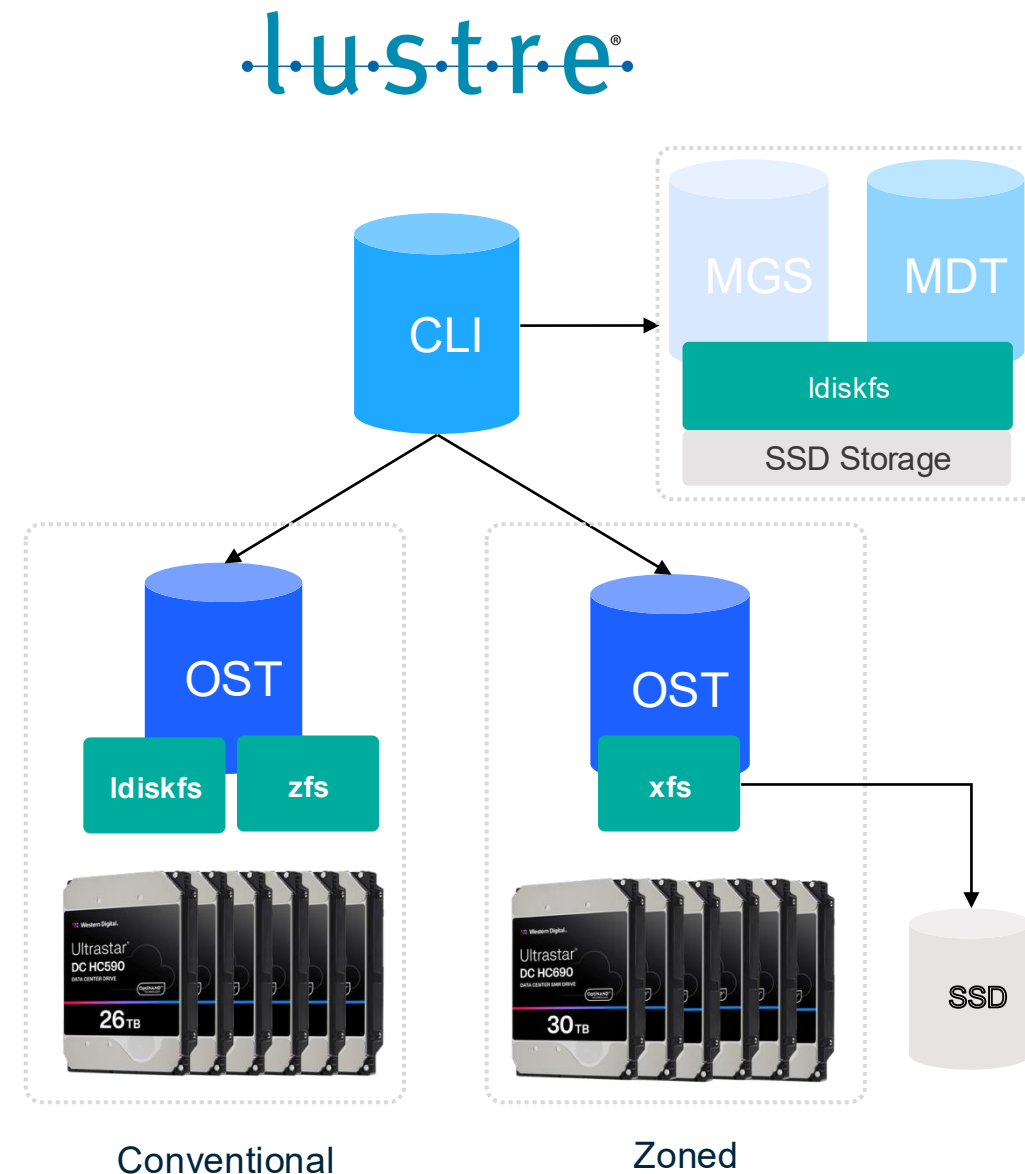
Evaluation

- **Software**

- Lustre: Main (dd48e6b096 – 7th July, 2026) + osd-xfs
- Kernel 7.0 + 1 patch (expose fiemap to be read by kernel users)
- Lustre cluster: 4 VMs (1xMGS/MDT/2xOSD/1xClient)
 - 16GiB memory for each
- OpenZFS: 2.4.4

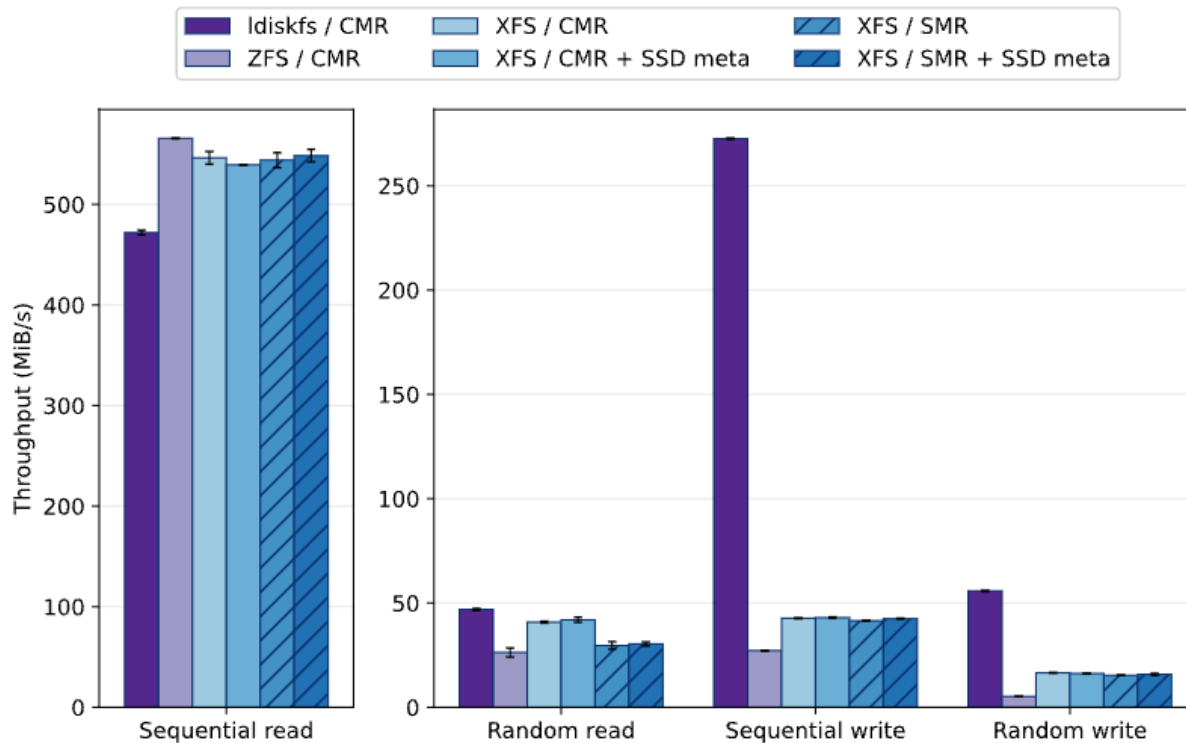
- **Hardware**

- 6x Western Digital UltraStar® HC670 26TB (CMR)
- 6x Western Digital UltraStar® DC690 30TB (SMR)
- Local SSD for XFS metadata in CMR/SMR + SSD configurations
- Dell R7515 /w 128GB Memory

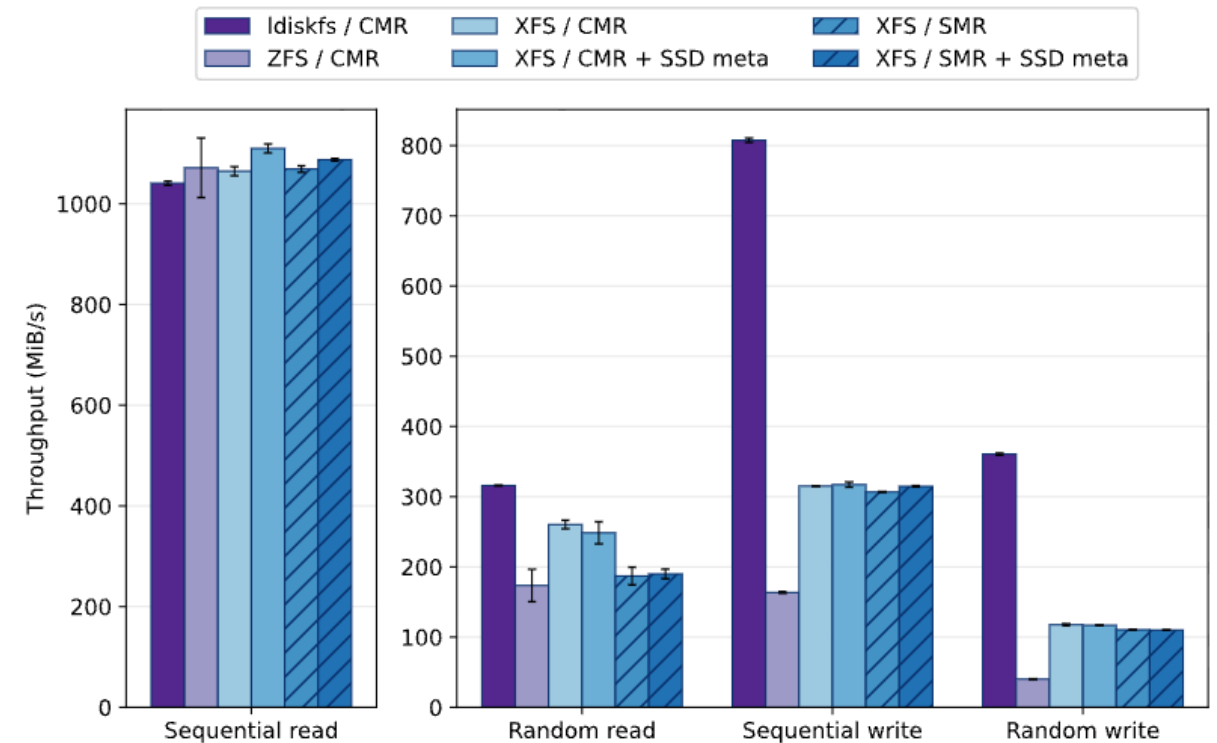


Small I/Os - 16 & 128 KiB IO Sizes

Lustre OST throughput at 16 KiB I/O, six OSTs, drive write cache disabled
Median of 2 x 15 s runs, error bars show min-max; 2026-09-25



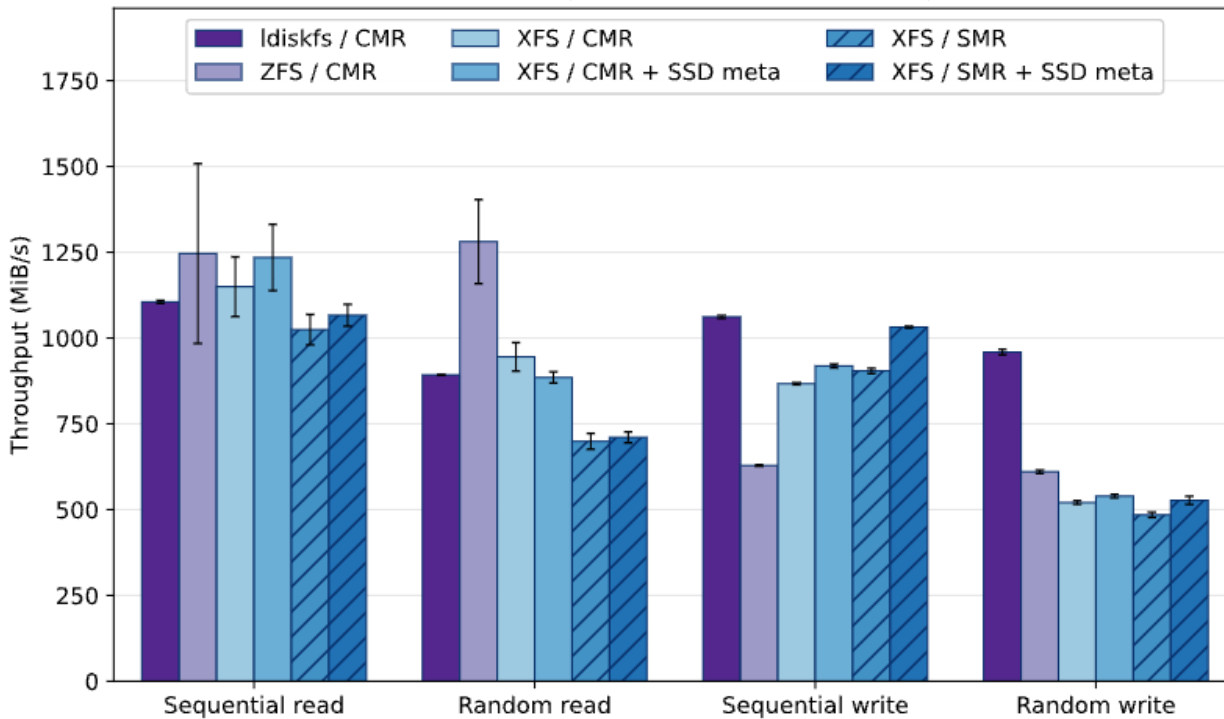
Lustre OST throughput at 128 KiB I/O, six OSTs, drive write cache disabled
Median of 2 x 15 s runs, error bars show min-max; 2026-09-25



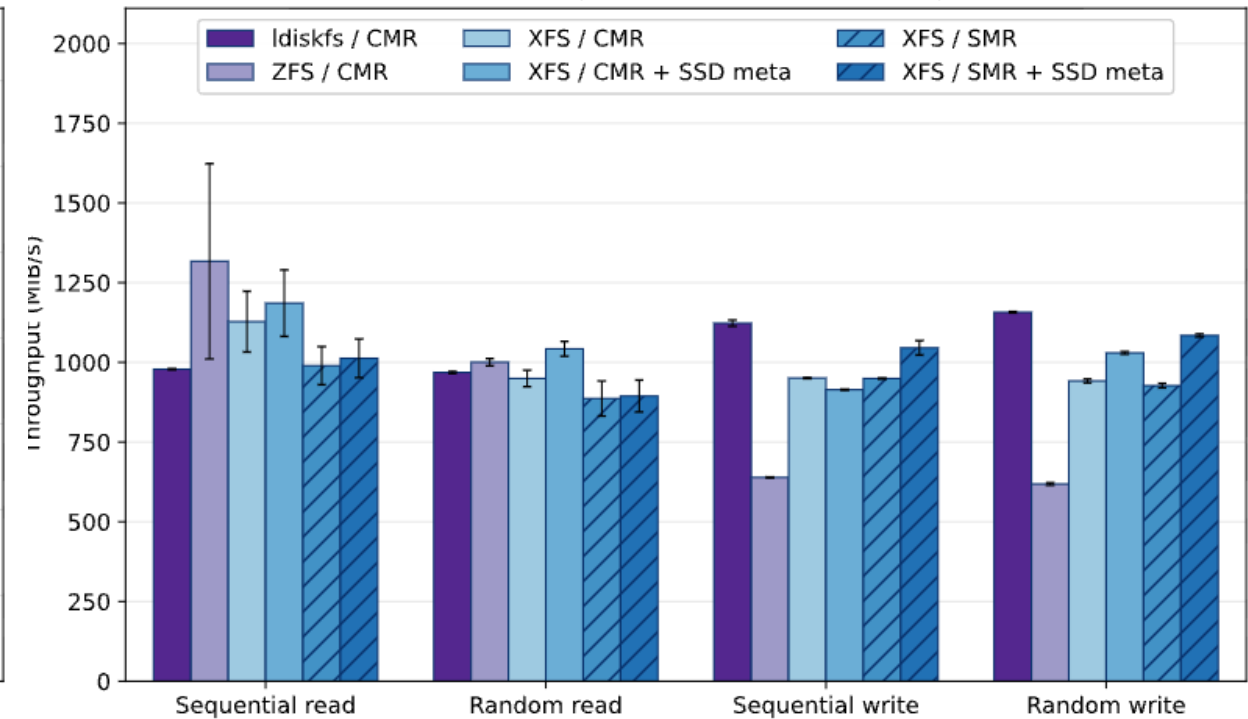
- Reads are within performance parity. Better out-of-box performance than ZFS.
- Large overhead on writes due to XFS Journal overheads. Optimizations being explored.

Large I/Os – 1 & 4 MiB IO Sizes

Lustre OST throughput at 1 MiB I/O, six OSTs, drive write cache disabled
Median of 2 x 15 s runs, error bars show min-max; 2026-09-25



Lustre OST throughput at 4 MiB I/O, six OSTs, drive write cache disabled
Median of 2 x 15 s runs, error bars show min-max; 2026-09-25



- Reads are similarly good. ZFS showing better performance
- Less overhead on writes due to less journal overhead.

Upcoming Features

1

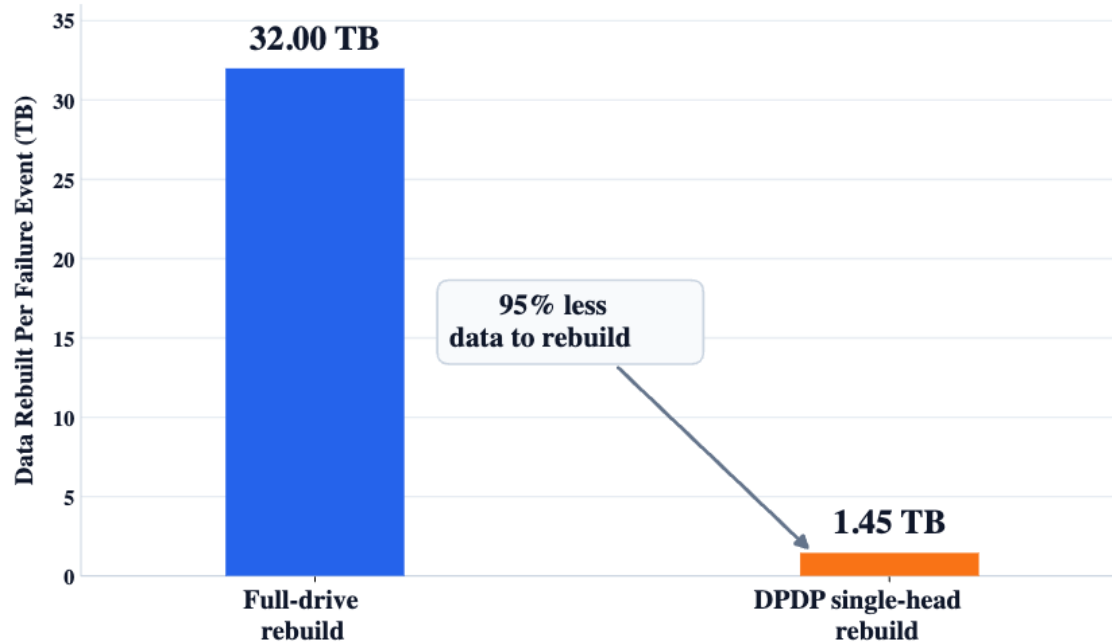
Data Preserving Depopulation (DPDP)

2

XRAID to XFS

SMR Enables Data Preserving Depop

Example 32TB, 11 Disk HDD



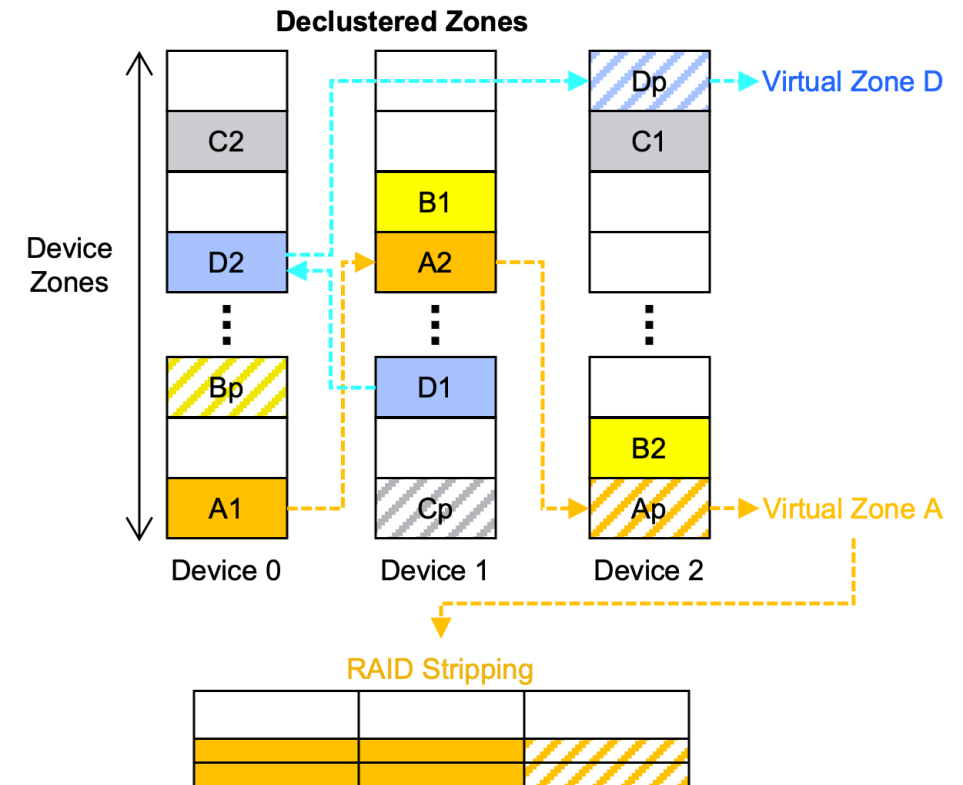
- **Head Depop Enables Keeping HDDs in the Field Longer**
- **SMR Head Depop Rebuilds Less Data Than CMR**
 - 100% SMR zones required (CMR zones data lost upon depop)
 - Repairs ~1.45 TB instead of the full 32 TB
 - Upon head depop event, `REPORT ZONES` will show affected zones with Zone Condition 0Dh (Offline)
- **~95% Reduction**
 - Massive savings in rebuild time and background I/O, leading to gains in system durability
- **Future-Proof**
 - Mitigate reliability risk with head failures (e.g. in HAMR)
 - As platter counts grow, the DPDP advantage gets even bigger

DPDP Kernel Integration

- We need several things to build on
 - Depopulation feature discovery and depopulation command support in SCSI/ATA
 - First batch of patches already queued for 7.3, more to follow
 - Kernel API and user ioctl interface to inspect and trigger HDD head depopulation
 - Notifications from the device driver to the file system of head depopulation completion
 - Likely through an FS registered block device callback (device operation)
- Mechanism for notifying a user of lost files already exist
 - Inotify/fanotify
- And of course, file system support
 - zonefs first (easier), XFS to follow

XFS RAID Support (SMR and CMR)

- C.f. session “RAID sucks (and what we can do about it)” by Christoph Hellwig
- Built on Zoned XFS (XFS SMR support)
 - Requires SSD for metadata with protected against device failures
 - E.g. RAID 1 SSD mirroring
- Uses append only “logical zones” built from a set of zones from different drives
 - Initial supports will have RAID 0, 5 and 6
 - Based on a declustered paradigm
 - Example: for 4D+1P RAID 5 configuration, one logical zone is built from 5 x 256 MiB zones, leading to 1 GiB logical zone

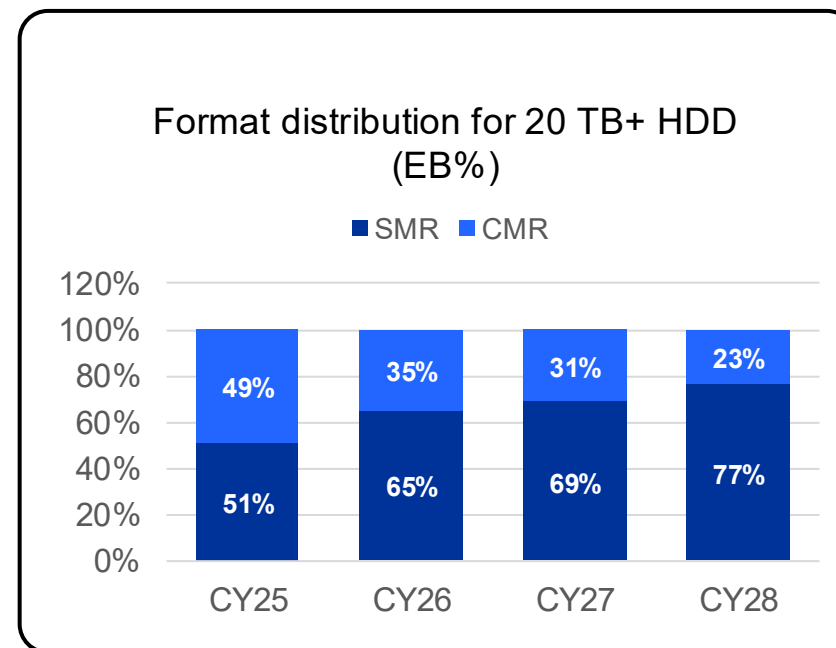


Status

- **A working PoC for Lustre that enables SMR HDDs through XFS**
 - There's still a write performance gap that needs to be addressed. Work with XFS developers to gauge feasibility (e.g., multi-object journaling).
 - Few XFS kernel changes are necessary
 - i.e., out-of-tree patches not to be maintained
 - Depop and XRAID features incoming to XFS
 - Eliminates the need for running mdraid and similar device mappers.
- **The elephant in the room**
 - As CMR volume shrinks, SMR HDDs will be the way forward for large-scale deployments.

• l u s t r e • + XFS®

Market Trends (EB TAM)



20TB+ estimated 88% of WDC EB in CY25 and 93% in CY27.
Sources: Estimated based on IDC Hard Disk Drive Forecast May 2026.



Multi-Object Optimizations in XFS

- Done, no XFS changes
 - Group commit: transactions are batched and share one log force. Lustre's commit callbacks run only after that force.
 - Recovery records as logged metadata: `last_rcvd` is stored in 512-byte XFS attributes, so the log force makes it durable.
 - Result: sequential write went from 588 to 672 MiB/s, compared to 825 MiB/s for `ldiskfs`.
- Prototype, requires XFS changes
 - Ordered-publication API (`xfs_ordered_*`): finishes the data first, then publishes the recovery record, followed by log checkout. This removes the extra data sync on every write.
 - Result: 648.5 to 766.5 MiB/s (+18.2%), with flushes per MiB roughly halved (0.485 to 0.204).
 - Not yet benchmarked with SMR
- Gaps
 - No multi-object crash atomicity: each VFS call commits its own XFS transaction. `ldiskfs` gets this from one journal handle per operation.
 - Intent log is diagnostic only: it records interrupted operations but never replays them.
 - No physical power-cut testing yet.
- Multi-object atomicity needs either substantial new code inside XFS or an OSD-side replaying intent log. Working with XFS maintainers if that's something we can do relatively easy in the XFS codebase.