Evolving XFS with Zoned Storage and Intelligent Data Placement

Hans Holmberg
Western Digital Research
hans.holmberg@wdc.com

Christoph Hellwig
Western Digital Research
christoph.hellwig@wdc.com

1 Introduction

We introduce Zoned XFS, which aims to deliver the high capacity, low wear, and high performance of zoned storage devices through the XFS enterprise filesystem [3, 7]. Zoned storage devices require host writes to be sequential within zones and force updates to made out of place. XFS was originally designed for in-place overwrites on conventional block devices, but was later enhanced to support out of place writes.

This poster abstract presents our efforts to enable zoned storage with XFS and how it expands the existing zoned storage work found in other enterprise filesystems [6] in two ways. Firstly, we utilize the zone append primitive [2], which eliminates the need to serialize the writes to the storage device. Secondly, we reduce write amplification and improve garbage collection performance by co-locating user data with similar lifetimes into zones based on file locality and application hints.

To achieve this, we enhance XFS with a new zoned allocator and a defragmenting garbage collection algorithm and evaluate the performance benefits on in-production storage devices using RocksDB [5].

2 Our approach and results

We implement zoned storage support based on the existing XFS real-time device feature [1], which enables using separate space allocators for metadata and file data. Metadata continues to use the existing B+ tree-based format requiring in-place updates on conventional storage. File data is allocated on zoned storage using our new zoned allocator. For file overwrites, the new data is redirected to newly allocated blocks using the existing mechanisms to support unsharing reference counted blocks (reflink)[8].

For file data placement, the new zoned allocator places data into a limited number of open zones utilizing the per-zone hardware write pointer to track available space, removing the need for persistent allocator data structures. Device writes are executed using the zone append [2] primitive which avoids the serialization that would be required when submitting regular write commands at the write pointer.

To reclaim the space occupied by blocks invalidated by file removal or overwrites, garbage collection is implemented. Valid data in partially used zones is located using the physical-to-logical block reverse mapping tree originally added to support online repair [9] and moved to new zones while performing automatic file data defragmentation. Because zones are filled with data of similar lifetime, we also reduce the amount of garbage collection that has to be performed.

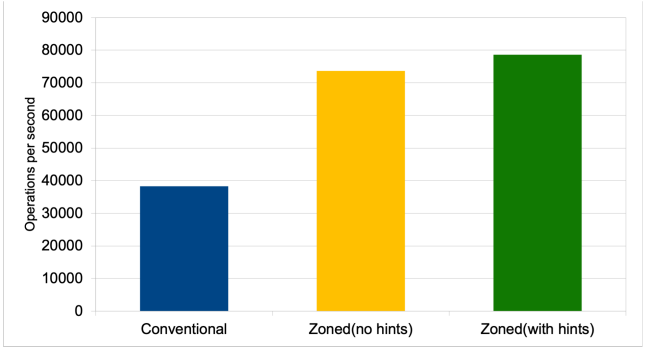

Figure 1. RocksDB db_bench overwrite benchmark, comparing XFS on conventional storage with Zoned XFS with and without write lifetime hint-based allocation.

We have evaluated the performance of the Zoned XFS implementation on production-based SSDs, comparing against baseline XFS on conventional SSDs using RocksDB [5] key-value store benchmarks and ZNS SSDs using the same hardware. Our experiments show that we can drastically reduce write amplification, improving a db_bench overwrite benchmark throughput by 92% by separating files into different zones. Utilizing write lifetime hints, where files of similar write lifetimes are co-located into the same zone, adds another 7%, see Figure 1. Our work is publicly available as open source and under review by the Linux community [4], opening up for further research in the area.

References

[1] Chandan Babu. XFS: Tracking space on a realtime device. https://blogs.oracle.com/linux/post/xfs-realtime-device, 2024.

[2] Matias Bjørling. Zone append: A new way of writing to zoned storage. Santa Clara, CA, February 2020. USENIX Association. https://www.usenix.org/conference/vault20/presentation/bjorling.

[3] Christoph Hellwig. XFS: The Big Storage File System for Linux. *USENIX ;login*, (34), 2009. https://www.usenix.org/system/files/login/articles/140-hellwig.pdf.

[4] Christoph Hellwig. RFC: support for zoned devices. https://www.spinics.net/lists/linux-xfs/msg93344.html, 2024.

[5] Meta Platforms, Inc. RocksDB: A persistent key value store for fast storage environments. https://rocksdb.org, 2025.

[6] Marta Rybczyńska. Btrfs on zoned block devices. https://lwn.net/Articles/853308, 2021.

[7] Adam Sweeney. Scalability in the XFS file system. In *USENIX 1996 Annual Technical Conference (USENIX ATC 96)*, San Diego, CA, January 1996. USENIX Association. https://www.usenix.org/legacy/publications/library/proceedings/sd96/full_papers/sweeney.txt.

[8] Darrick Wong. XFS - Data Block Sharing (Reflink). https://blogs.oracle.com/linux/post/xfs-data-block-sharing-reflink, 2020.

[9] Darrick Wong. XFS - Online Filesystem Repair. https://blogs.oracle.com/linux/post/xfs-online-filesystem-repair, 2024.