

Whereas rebuild/replacement of failed RAIDZ or RAID5/6 devices is limited to adjacent peers dRAID is able to pull from all the block devices that make up a virtual device and enlist their aggregate bandwidth to replace or rebuild failed devices.

Conclusion

Enlarge

Enlarge

RAIDZ1, 2, 3 - Like RAID5, both data and error-correcting parity blocks are striped across a "virtual device" (vdev). The complete data record is broken up into chunks and spread across "N" number of data devices and 1, 2, or 3 "parity" blocks ("P" blocks below). Both RAIDZ and RAID5/6 generally support dedicated spares that can, in software, be activated to replace a failing

primary device.

Enlarge

Mirror: like RAID 1, data is duplicated across multiple devices. The data is written across multiple devices, ensuring that as long as one copy is readable, the data is retrievable. Unlike many RAID1

implementations where two or more data sources disagree, ZFS

and its checksums know which copy is correct and which contains

errors. Mirrors can offer the full performance of a single device, but

not more (unlike stripes which can leverage the concurrency of

multiple devices).

Enlarge

0 Diek DAIDE / DAID74

devices. This does not offer any redundancy or data protection (other than the detection of corruption), but for temporary scratch space and for ephemeral data, offers the highest performance.

 \bigoplus Enlarge

RAIDZ2, RAID6 builds on the concept of distributed parity, with additional parity blocks (the P1/P2 blocks below). Unlike RAID6,

RAIDZ can support up to three parity blocks per data stripe

(RAIDZ3).

G-P1

Enlarge

Virtual devices (VDEVs) are …. and typically created by combining x number of disks. There is a practical upper limit of disks with both dRAID or ZRAID virtual devices. As we increase the number of disks that make up a VDEV, user data has to be "chunked" evenly across all participating member disks.

In a 16 data plus 2 parity scenario, a 1MiB object write (or record) translates into each of the 16 disks being given a 16K "chunk" of the original data. This works fine, but if the data object is only 16K bytes, there is not an efficient way to subdivide it up evenly across all participating disks. ZRAID has clever ways to pack "records" but dRAID does not, which means write efficiency drops as more disks are added to a virtual device.

On read, the problem is compounded by the fact that a stripe or record can not be reconstructed until all disks have contributed their chunks. Aggregate read latency is a function of the slowest device to respond. As we add more disks to a virtual device, read operations tend to have higher completion latency.

Challenges with Using VDEVs to Create Very Large File Systems.

Once a disk group is created, it can be partitioned into up to 1,024 logical block volumes. There are several reasons one may wish to do that, but for OpenZFS applications, we found no performance advantages over a single large logical volume that encompassed the entire disk group.

 $E_{\rm eff}$ if we figure doubt a way to "stack" $Z_{\rm eff}$ values on top of one another such that such that such that rather than striping vdevs, we could construct a ZRAID vdev of several child ZRAID vadevs, we still have the problem of managing all those individual disks from all those individual disks from a single host.

The Exos Corva Π enables us to transform up to 106 disks into a pair of defect-only a pair of defect-only a pair of defect-only at Γ free, self-healing logical volumes that serve as "disks" when composing ZFS vdevs. Where once the host server might have had to manage 106 independent disks, each with a capacity of tens of tens of tens of TeraBytes (TB), it now only sees two very large block \mathbf{H} device volumes, each with a capacity of 1 PB or more. The capacity of 1 PB or more. The capacity of 1 PB or mor

It's relevant to point out here that in transforming a disk group of up to 53 drives into a single logical block volume, the host no longer has to manage those disks individually but rather is presented with a single reliable block device with a capacity of a petabyte or more.

Further, since the Exos CORVAULT has two controllers, a total of up to 106 disks can be used to create two logical volumes, each of 1 PiB or more of capacity. Exos CORVAULT also brings a high-performance, multi-gigabyte intelligent RAM caching layer that can be treated as non-volatile with both mirroring and flash+ultra-capacitor power loss protection that enables read-ahead and write caching in ways no host server could orchestrate with volatile host-based DRAM alone.

It should also be noted that in the event of a controller failure, either controller can take over the disk groups of the other controller while maintenance is performed, while the host maintains access via the partner controller using well-defined multipath hardware and software features.

At the time of \mathbf{H} allocates values on \mathbf{H} each device based on the discovered capacity of the top-level vdev. Because the zpool capacity presented by 8 Exos CORVAULT is over 10 PiB, the default metaslab allocation algorithm would create far more metaslabs than necessary, resulting in wasted disk space. After much investigation and tuning, limiting the number of number of number of number of n metaslabs to 2048 is recommended. This must be set before creating the zpool. Failure to limit the metaslab allocation count can hurt overall performance and overall space utilization.

echo 2048 \pm /sys/modules/zfs/parameters/ \pm /sys/modules/zfs/parameters/ \pm

The answer to the challenge with current VDEV configurations is to stripe multiple virtual devices together.

 $\mathcal{L}_{\mathcal{A}}$ raidz1 cv1-a cv2-a cv3-a cv4-a cv5-a cv6-a cv7-a cv8-a \ raidz1 cv1-b cv2-b cv3-b cv4-b cv5-b cv6-b cv7-b cv8-b

 z pool add DESTOR -f -o ashift=12 special $\mathbf 1$ /dev/nvment1 $\mathbf 1$ /dev/nvment1 $\mathbf 1$ mirror /dev/nvme2n2 /dev/nvme3n3

The Exos CORVAULT platform internally implements an algorithm in hardware that is conceptually identical to either dRAID2 or dRAID3 above, but it does so without requiring any host compute resource, by using the resources within the Exos CORVAULT platform itself to perform the erasure coding across the disks.. Via the native REST, WebUI, or CLI, an administrator can combine disks into either 8+2, 16+2, or 16+3 "disk groups" where 8 or 16 is the number of data "chunks" per either 2 or 3 parity "chunks".

The following command creates a zero specific parameters a \mathbb{R}^n specific parameters for \mathbb{R}^n size and metadata. The primary calculation instructs $\mathbb H$ metadata in its primary (ARC) cache. The redundant metadata in its primary (ARC) cache. The redundant metadata
Metadata stores and an extra copy of metadata for additional protection and improvement of Γ random I/O.

The zpool configuration for data protection and space utilization is shown below. For data protection is shown below. For data protection Γ maximum data protection against individual controllers or $\mathbf H$. The entire Exos CoRVAULT controllers or entire failures, the best practice is to identify and separate the volumes served by the A and B controllers. Volumes owned by each Exos CORVAULT' A controller will be grouped into one VDEV, and the vOEV, and the volumes owned by the $\rm H_{\odot}$ controllers will be grouped into a set separate VDEV. Both of the A/B VDEVS will be combined into a single zpool along with the addition of a special mirrored value of a special mirrored value Π contain the data set $\mathbb R$ on mirrored SSDs.

Here, we illustrate with an example that builds on the previous dRAID visualizations and lays it out linearly across 53 disks. Each "chunk" of data is distributed as eight data blocks and two parity blocks. Additionally, the Exos CORVAULT disk group includes reserved spare capacity on each disk that is used for repair (due to disk failure). As such, each data stripe occupies blocks on 10 disks. The boundary of each logical block of 10 chunks is logically adjacent to the next logical block with dRAID2, but Exos CORVAULT leverages a pseudo-random distribution of stripe groups across groups of disks to ensure even distribution of data across all of the disks that are participating in the disk group. This offers numerous benefits, including: • Flexibility to create disk-groups with larger numbers of included disks,

 $Z\in\mathbb{R}^n$ contains a variant parameters that can overwhelm even for the tuning parameters that can overwhelm even for the tuning parameters that can overwhelm even for the tuning parameters of the tuning parameters $\$ most seasoned storage professional. Many experts are active in the continued development care of OpenZFS. Seagate has consulted with the OpenZFS community and partners are defined with Klara Inc. to identify specific Γ module parameters Γ and additional settings that take advantage of Exos CORVAULT-specific features and the large LUN sizes recommended for the LUN sizes recommended for the Web3 use case. As a result of this t activity, the recommended tuning parameters are shown below. For those interested tuning parameters are shown b in the details of each parameter, documentation can be found here: <https://openzfs.github.io/openzfs-docs/Performance%20and%20Tuning/index.html>

The specific optimizations Γ optimizations Γ . The upstream OpenZFS 2.2 to ensure optimal Γ support for Exos CORVAULT systems are detailed here: [https://github.com/suykerbuyk/disk-helpers-scripts/blob/main/](https://github.com/suykerbuyk/disk-helpers-scripts/blob/main/Seagate.Sponsored.Zfs.Changes.txt)

These settings can be applied directly each module \mathbf{I}_c individual files for temporary \mathbf{I}_c use. To make these parameters persistent, create a zfs.conf file in the /etc/

models are the contributions of the contributions of the contributions of the contributions of the contributions

independent of the erasure coding geometry (8+2, 16+2 or 16+3). • Faster rebuilds after disk failure due to all disks contributing spare capacity • Support for mixed capacity disks within the disk-group due to the distributed nature of the data, parity and spare chunks across all disks within the disk-group • Support for advanced self-healing features such as Autonomous Drive Regeneration (ADR).

Stripe/Block 0 Stripe/Block 1 Stripe/Block 2 Stripe/Block 3

Stripe/Block 5 Stripe/Block 6 Stripe/Block 7 Stripe/Block 4 P1 S0 D0 D1 D2 D3 D4 D5 D7 P0

Stripe/Block 8 Stripe/Block 9 Stripe/Block 10 Stripe/Block 11 $\frac{15}{15}$
 $\frac{15}{17}$

 \bigoplus Enlarge

zfs_max_recordsize=16777216 options zfs zfs_vdev_max_ms_shift=40 options zfs zfs_vdev_def_queue_depth=256 options ω as ω as ω as ω as ω as ω as ω options ω options ω

 $z=\pm\sqrt{2}$ options $z=\pm\sqrt{2}$ options $z=\pm\sqrt{2}$ $z=\pm\sqrt{2}$ options $z=\pm\sqrt{2}$ options $z=\pm\sqrt{2}$

options zfs metaslabalistics metaslabalistics metaslabalistics options zfs metaslabalistics options zfs metas

 $z=\omega$ active ω options z active ω options z active ω options z active ω

$\sigma_\mathrm{max}=15$ and $\sigma_\mathrm{max}=515$ and $\sigma_\mathrm{max}=515$

The Exos CORVAULT Platform Provides an Advanced VDEV Solution and Compute Offload Engine

The following settings are following are for the SPL module and can be put into and can be put into an spl.conf similar to the above for persistence after reboots.

options spl spl_kmem_cache_kmem_threads=8 options spl splay species that can be splitted in the caches splitter $\mathbf{I}_\mathbf{z}$ operations splitter in the splitter $\mathbf{I}_\mathbf{z}$

The diagram shows that each Exos CoRVAULT is split into two large protections Γ pools or disk groups and volumes and assigned to a separate controller. The Linux host needs enough SAS ports to connect to each Exos CORVAULT controller (2 per Exos CORVAULT). ZFS uses each LUN to build a zpool that provides additional data protection using one LUN from each controller group.

To accommodate this scale, the host uses 16 SAS ports (4 x 4-port HBA's) with one \mathbb{R} connection into each of the Exos CoRVAULT A/B controllers as shown into \mathbb{H} controllers as shown in

We can continue to scale "up" the capacity by adding more Exos CORVAULTs to the solution. With two SAS targets per Exos CORVAULT, a stack of 8 CORVAULTs appears to be 16 very large disk targets to the host.

There is minimal performance degradation as experienced by the attached host \mathbb{R}^n accessing the Exos CORVAULT volumes during single drive rebuild since each disk with the disk-group is participating in both the read and write and write and write and write and write activity necessary to \sim complete the repair. This is contrasted to traditional RAID5/RAIDZ1 or RAID6/ RAIDZ2 implementations where the repair performance is constrained by how quickly the system can write to a single spare/replacement disk. By leveraging distributed spare capacity reserved on each disk, CORVAULT completes repair activity in less time, and with less perceived performance impact to the attached host

The architecture described in this paper using eight Exos CoRVAULT arrays with Exos CORVAULT arrays with Exos C \mathbb{P}^2 provides over 12x 9's over 12x 9's of data durability in addition to point-in-time in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time-in-time consistency. ZFS employs a little state copy-on-write, point-in-time consistent model such that no file system check and

We also accomplished something else in layering ZFS on top of Exos CORVAULT. S ince C ORVAULT implements something very similar to d R ID internally, the top-dRAID internal ly, the top-d level ZPOOL is a product of layering erasure coding protected vdevs. Our "9's" of durability add together. If Exos CORVAULT offers five "9's" of durability and a given Z -9's in the aggregate, the aggregate, they of durability $\mathbf{1}_{1}$ "9's of durability $\mathbf{1}_{2}$ -9's of durability $\mathbf{1}_{3}$ a number that is largely unachievable without resorting to complex and expensive network clustered file systems and all of the additional compute nodes required. This implementation, often described as "multi-level erasure-coding", offers numerous benefits which work together to result in higher durability for the overall solution.

The following command combines 2 raidz1 (7+1) vdevs into a large zpool named DESTOR that includes a special metadata device on NVMe SSD (nvme1n1p1). The actual volume names in this example have been replaced with readable names that correlate to the A or B controller on each CORVUALT. It is highly recommended to use the /dev/disk/by-id/wwn-0x names for zpool creation as these are fixed. Using / dev/sdX or /dev/nvme* names are not always persistent across reboots. The notation across reboots. The notatio zpool create DESTOR -O recordsize=1M -O atime=off -O dnodesize=auto -o

Add in the special metadata device mirrors:

zfs create -o recordsize=1M -o primarycache=metadata -o redundant_metadata=most DESTOR/<DATA-SET>

Rigorous testing has found the optimal ZFS record size using 8 Exos CORVAULT

arrays to be 1M.

Specific settings at zpool creation time

Creating the Zpool and ZFS data set

Recommended ZFS and OS Tuning: The Reference Design for OpenZFS on **CORVAULT** Hyperscaling OpenZFS with EXOS **CORVAULT** The CORVAULT Platform Provides an Advanced VDEV Solution and Compute Offload Engine Why OpenZFS has become the default file system for Filecoin **Introduction**

Conclusion

[Seagate.Sponsored.Zfs.Changes.txt](https://github.com/suykerbuyk/disk-helpers-scripts/blob/main/Seagate.Sponsored.Zfs.Changes.txt)

 $Z_{\rm eff}$ specific tuning parameters:

Setting max_sector_kb to 8192K (8MB) – this to match Exos CORVAULT aggregated LUNS.

$\mathcal{L}^{\text{max}}_{\text{max}}$

echo 8192 > /sys/block/(device)/queue/max_sectors_kb

options zfs vdev_ms_count_limit=8192 options zfs zfs_vdev_aggregation_limit=16777216 options zfs

Recommended ZFS and OS Tuning:

The following table shows the configuration components used for this reference architecture.

the diagram below:

A high-level Block diagram is shown below.

A multi-level erasure-coded architecture provides data durability far beyond a singlelevel scheme. Additional benefits of this model are highly improved rebuild times of individual drive failures that reduce exposure to additional failures. But perhaps the greatest benefit is the offloading of the offloading of the host CPU for rebuild cycles. In definition, Γ failure detection and recovery are off-loaded to the VelosCT ASIC within each Exos \mathbf{H} controller, which implements the ADAPT distributed erasure distributed e coding algorithm.

repair is ever needed. While this document shows how 8 Exos CoRVAULT arrays how 8 Exos CoRVAULT arrays can provide massive massive ma capacity using OpenZFS, the same principles can be applied when using more or less arrays. Seagate technical marketing plans to continue building, testing, and

publishing best practices using OpenZFS on a single or multiple Exos CORVAULT arrays with this Multi-level EC model.

A Multi-Level Erasure Code Architecture:

- 1) Eight Seagate Exos CORVAULTs a) Each populated with 106 Seagate Exos 16 TB drives.
- b) 848 total disks.
- c) Two 53 disk, disk groups per Exos CORVAULT in a 16+2 + 10% spare configuration. d) is exported as a single state group is exported as a single SAS target/volume. The single SAS target/volume
- 2) Society (2) Server Society (2) Server Server Server a) 18 x 2 cores
- b) 256 GB RAM c) and the LSI Broadcom 9405-16E HBAs and the LSI Broadcom 9405-16E HBAs and the LSI Broadcom 9405-16E HBAs and
- 3) Special Meta Data devices
- a) 4 each, 3.84 TB NVMe drives in a striped mirror.

when compared to traditional data protection schemes. The compared to tradition schemes. The comparison scheme

The Bill of Materials

The Reference Design for OpenZFS on Exos CORVAULT

Performance modeling for the Filecoin Storage Miner use case was accomplished using the FIO benchmarking tool. Profiles for both storing sectors (writing $32g$ files) \sim and proving (random reads of 16K sections of those 32 GB sectors) were run in parallel. The results below show that the combination of kernel and ZFS module tuning helped increase both streaming write and small random reads in $\rm H$ random reads in some $\rm H$ **Modeling for the Filecoin Storage Miner use-case**

cases 3x.

Performance Testing

A multi-level erasure code architecture can provide data durability, consistency, and computational efficiency for the petabyte-scale storage systems needed for decentralized Storage Providers. With the combination of Exos CORVAULT and OpenZFS, Seagate is leading the way with cost-effective, easy-to-implement, and maintainable solutions. Seagate will continue to foster the development of architectures and other best practices around the configuration of Exos CORVAULT with OpenZFS.

This section provides a basic foundation for disk-based aggregation and data protection implementations that have long been used when creating large data

storage solutions comprised of multiple individual disk drives. The goal of these solutions is usually twofold: 1. To aggregate the capacity and performance of multiple individual drives into a

larger logical group which is easier to manage. 2. To create a solution that is tolerant to failure of one or more individual

components, such as a disk.

High level descriptions of traditional disk-based aggregation and data protection implementations are provided here to improve the understanding of the reader when we later discuss the more advanced distributed multi-level erasure-coding implementation in this paper.

ZFS Topologies and their RAID analogies

For the reasons stated above, OpenZFS has quickly become the default file system for the Filecoin network. The OpenZFS file system ensures, above all else, end-toend data integrity.

CRC checksums are used to detect silent data corruption with every IO operation to and from the storage subsystems to the data landing in host memory.

The evolution of OpenZFS also post-dates most conventional RAID topologies and so many of these prior solutions – which promised to be the answer to solve the biggest data challenges – do not take full advantage of ZFS capabilities.

OpenZFS is making great strides in fulfilling this promise for direct-attached storage. OpenZFS can offer the functionality of most legacy RAID systems but in ways that are not bound by the fixed topologies of traditional RAID configurations. One example is the ability to detect silent data corruption and silently perform inline data repairs, in line. Another is offering the storage architect the ability to construct VDEVs stripes that load balance IOPs.

Why OpenZFS has become the default file system for Filecoin

OpenZFS is an advanced open-source file system and volume manager technology designed for managing and protecting large amounts of data. It originated from the ZFS (Zettabyte File System) project, which was initially developed by Sun Microsystems and later open-sourced by Oracle. OpenZFS was created as a community-driven effort to continue the development and improvement of ZFS independent of Oracle.

OpenZFS is increasingly used in enterprise and commercial-grade environments due to its robust features and capabilities. Of particular note are accommodations to address the specific storage needs and data integrity requirements of large, onpremise data infrastructure. Today, you'll find it used to implement everything from most SMB NAS systems to the backing storage for the world's largest highperformance super-computer Lustre file systems.

OpenZFS

The CORVAULT family of Seagate platforms combines certain key ideas learned from software-defined storage platforms such as OpenZFS, CEPH, and Lustre and combines them with Seagate's unique insight into the optimal operational characteristics of individual drives. As such, the Exos CORVAULT platform is uniquely qualified to serve as a compute-offload engine for massive, direct-attached disk arrays.

This platform combines the best of proven software-defined storage algorithms with Seagate's intrinsic domain expertise of hard drives to maximize longevity and performance in ways no third party can do with generic device models.

As all the disk aggregation and management is done in hardware, the host is no longer bound by how many disks it can manage and still performs useful work. The Exos CORVAULT platform solves the scaling problem by performing disk aggregation, defect management, and cache management at the enclosure controller level instead of relying upon host compute and IO resources in other solutions. The resulting output is storage in the form of very large block volumes. The benefit of this approach is that compute resources per petabyte are dramatically reduced, as is the complexity of the IO interconnects.

Seagate Exos CORVAULT

Introduction

Filecoin storage is a very well-structured storage format as a consequence of the zero-knowledge cryptographic algorithms that underpin its immutable proofs of unique replication across space and time. This facility allows for globally distributed, verifiable, cryptographically unique replicas across network nodes in a cost-effective and performant manner. Improvements can be made, however, using existing file systems in combination with advanced storage hardware to gain even greater performance and resilience.

The Zetabyte File system (ZFS) is unique in that it has a 20-year history that demonstrates the ability to scale into the 10s of petabytes efficiently range for directattached storage solutions.

Combining the OpenZFS protocol with very large defect-free, SAN-like volumes from Exos CORVAULT presents a unique opportunity for the decentralized storage ecosystem to reduce compute infrastructure by a factor of four or more all while increasing resiliency in the process. This combination will reduce costs for storage providers in the Filecoin network and accommodate commercial-grade storage loads for existing web2 applications and future AI or web3 data loads.

Executive Summary

Performance Testing

Table of Contents

EXOS **CORVAULT** S SEAGATE

MOZQiC 3+

White Paper

or specifications. CS58.1-2404US