ZFS 101—Understanding ZFS storage and performance

Enlarge / No, you can’t actually buy Ironwolf disks with an OpenZFS logo on them—but since they’re guaranteed SMR-free, they are a solid choice.
Jim Salter

reader comments

197 with 87 posters participating, including story author

Share this story

Storage fundamentals

View more stories

As we all enter month three of the COVID-19 pandemic and look for new projects to keep us engaged (*read*: sane), can we interest you in learning the fundamentals of computer storage? Quietly this spring, we’ve already gone over some necessary basics like how to test the speed of your disks and what the heck RAID is. In the second of those stories, we even promised a follow-up exploring the performance of various multiple-disk topologies in ZFS, the next-gen filesystem you have heard about because of its appearances everywhere from Apple to Ubuntu.

Well, today is the day to explore, ZFS-curious readers. Just know up front that in the understated words of OpenZFS developer Matt Ahrens, “it’s really complicated.”

But before we get to the numbers—and they are coming, I

promise!—for all the ways you can shape eight disks' worth of ZFS, we need to talk about *how* ZFS stores your data on-disk in the first place.

Zpools, vdevs, and devices

- This full pool diagram includes one of each of the three support vdev classes, and four RAIDz2 storage vdevs.
- You wouldn't typically want to make a "mutt" pool of mismatched vdev types and sizes—but nothing's stopping you, if that's what you want to do.

To really understand ZFS, you need to pay real attention to its actual structure. ZFS merges the traditional volume management and filesystem layers, and it uses a copy-on-write transactional mechanism—both of these mean the system is very structurally different than conventional filesystems and RAID arrays. The first set of major building blocks to understand are zpools, vdevs, and devices.

zpool

The zpool is the uppermost ZFS structure. A zpool contains one or more vdevs, each of which in turn contains one or more devices. Zpools are self-contained units—one physical computer may have two or more separate zpools on it, but each is entirely independent of any others. Zpools cannot share vdevs with one another.

ZFS redundancy is at the vdev level, not the zpool level. There is absolutely *no* redundancy at the zpool level—if any storage vdev or SPECIAL vdev is lost, the entire zpool is lost with it.

Modern zpools can survive the loss of a CACHE or LOG vdev—though they may lose a small amount of dirty data, if they lose a LOG vdev during a power outage or system crash.

It is a common misconception that ZFS “stripes” writes across the pool—but this is inaccurate. A zpool is not a funny-looking RAID0—it’s a funny-looking JBOD, with a complex distribution mechanism subject to change.

For the most part, writes are distributed across available vdevs in accordance to their available free space, so that all vdevs will theoretically become full at the same time. In more recent versions of ZFS, vdev utilization may also be taken into account—if one vdev is significantly busier than another (ex: due to read load), it may be skipped temporarily for write despite having the highest ratio of free space available.

The utilization awareness mechanism built into modern ZFS write distribution methods can decrease latency and increase throughput during periods of unusually high load—but it should not be mistaken for *carte blanche* to mix slow rust disks and fast SSDs willy-nilly in the same pool. Such a mismatched pool will still generally perform as though it were entirely composed of the slowest device present.

vdev

Each zpool consists of one or more vdevs(short for virtual device). Each vdev, in turn, consists of one or more real devices. Most vdevs are used for plain storage, but several special support classes of vdev exist as well—including CACHE, LOG, and SPECIAL. Each of these vdev types can offer one of five topologies—single-device, RAIDz1, RAIDz2, RAIDz3, or mirror.

RAIDz1, RAIDz2, and RAIDz3 are special varieties of what storage greybeards call “diagonal parity RAID.” The 1, 2, and 3 refer to how many parity blocks are allocated to each data stripe. Rather than having entire disks dedicated to parity, RAIDz vdevs distribute that parity semi-evenly across the disks. A RAIDz array can lose as many disks as it has parity

blocks; if it loses another, it fails, and takes the zpool down with it.

Mirror vdevs are precisely what they sound like—in a mirror vdev, each block is stored on every device in the vdev. Although two-wide mirrors are the most common, a mirror vdev can contain any arbitrary number of devices—three-way are common in larger setups for the higher read performance and fault resistance. A mirror vdev can survive any failure, so long as at least one device in the vdev remains healthy.

Single-device vdevs are also just what they sound like—and they're inherently dangerous. A single-device vdev cannot survive any failure—and if it's being used as a storage or SPECIAL vdev, its failure will take the entire zpool down with it. Be very, very careful here.

CACHE, LOG, and SPECIAL vdevs can be created using any of the above topologies—but remember, loss of a SPECIAL vdev means loss of the pool, so redundant topology is strongly encouraged.

https://arstechnica.com/?p=1672365