OpenZFS for storing Science Data

Initial Thoughts on Using OpenZFS as Block Storage for Scientific Data

Tino Reichardt

Nikhef Amsterdam, Mai 7, 2026

HELMHOLTZ

DESY.

Overview

Why OpenZFS?

What are the points of OpenZFS within Science

Version numbers

OpenZFS within the main Linux distributions
Releases and their new features

What are those vdevs

What are the points of OpenZFS within Science

1. Uncompromising Data Integrity
   - End-to-End Checksumming + Self-Healing + Protection against Bit Rot
2. Scalability and Performance for Big Data
   - Capacity: OpenZFS uses 128-bit addressing
   - Advanced Caching via various vdev types: SLOG, DDT, L2ARC, SPECIAL
3. Data Management and Reproducibility
   - Instant and cheap Snapshots and Efficient Cloning with Promoting
4. Cost Efficiency and Flexibility
   - Built-in Compression and no dependency on proprietary RAID controllers
5. Security and Data Protection
   - Encryption and Ransomware Mitigation because of its copy-on-write nature

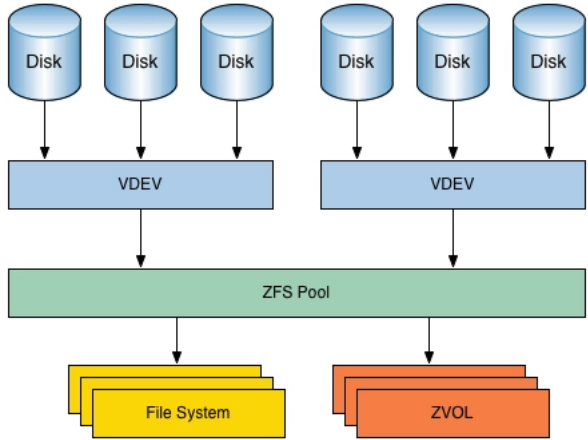
OpenZFS within the main Linux distributions

- Ubuntu 24.04 LTS: OpenZFS 2.2.x, EOL: May 2029
- Ubuntu 26.04 LTS: OpenZFS 2.4.x, EOL: May 2031
- Debian 12 (Bookworm): OpenZFS 2.1.x, EOL Jun 2028
- Debian 13 (Trixie): OpenZFS 2.3.x, EOL Jun 2030
- Debian 14 (Forky): OpenZFS 2.4.x, EOL Jun 2032
- RedHat Linux, AlmaLinux, RockyLinux
  - extra repository needed, but not difficult to use
  - two options, DKMS and prebuilt Kernel modules

Releases and their new features

- zfs-2.4.0 (2025-10)
  - new dedup optimizations, block cloning optimizations
  - zfs rewrite, ZIL can be on special vdev
- zfs-2.3.0 (2025-01)
  - New fast dedup, Direct IO, JSON, RAIDZ Expansion
  - replaced Buildbot with Github Actions CI
    - Tests: AlmaLinux, CentOS, Debian, Fedora, FreeBSD, Ubuntu
- zfs-2.2.0 (2023-10)
  - Block cloning, Linux container support, Vdev properties, ZSTD early abort
  - Checksums: Hardware acceleration for SHA2, new type: BLAKE3 and Edon-R cleanups
- zfs-2.1.0 (2021-07)
  - Distributed Spare RAID (dRAID)

OpenZFS / Vdev overview



Vdev hierarchy

ZFS has around XXX ...

- poolname (root vdev)
  - raidz1-0 (top-level vdev)
    - /dev/sda1 (leaf vdev)
    - /dev/sdb1 (leaf vdev)
    - /dev/sdc1 (leaf vdev)
    - /dev/sdd1 (leaf vdev)
  - raidz1-1 (top-level vdev)
    - /dev/sde1 (leaf vdev)
    - /dev/sdf1 (leaf vdev)
    - /dev/sdg1 (leaf vdev)
    - /dev/sdh1 (leaf vdev)
- top level vdevs: you need all of them and IOPs scale linear with the count of them
- leaf vdevs can be replaced with spare or directly
- the manpage `zpoolconcepts(8)` has an good overview

Default data vdevs types

- single disk
- mirror (RAID1) - double read speed, single write
- raidz1 (RAID5), raidz2 (RAID6), raidz3 (triple parity)
  - variable stripe width, no write hole
  - data and parity is striped across all disks within a vdev
- draidN (N times RaidZN)
  - like multiple raidzN with integrated distributed hot spares for faster resilvering
  - uses a fixed stripe width (4KiB x 8 disks this will be 32KiB, even for files of 3 KiB)
  - draid[parity][:datad][:spares s][:children c]

dRAID is it's own thing

- draid[parity][:datad][:spares][:childrenc]
  - A non-default dRAID configuration can be specified by appending one or more of the following optional arguments to the draid keyword:
  - parity - parity level: 1-3, Default:2
  - data - number of data devices per redundancy group Default:8
  - spares - number of distributed hot spares, Default:0
  - children - expected number of children
- typical setting is like this: draid2:8d:3s:XXc

Unlike RAIDZ, dRAID cannot do partial-stripe writes efficiently, small blocks are padded with zeros to meet the fixed stripe width, this is: wasting space! Therefore the Special vdev Type was intruduced.

Vdevs with special functionality

- Spare: a drive that acts as a hot spare, automatically replacing a failed drive in another vdev
- Cache (L2ARC): a Level 2 ARC vdev used for caching frequently accessed data to improve random read performance
- SLOG: separate log vdev (SLOG): store the ZFS Intent Log (ZIL) which improves synchronous write performance
- Dedup: a vdev dedicated strictly to storing the Deduplication Table (DDT)
- Special: store metadata, optional: small_file_blocks, DDT and SLOG since 2.4.x

Sample calculations for a disk with 100 IOPs and 200 MiB/s.

Vdev type	Vdev count	Useable space	IOPS-RD	IOPS-WR	MiB/s WR	MiB/s WR
Single (one disk)	1	100	100	100	200	200
Single (10x Single)	10	100	1000	1000	2000	2000
Stripe (1x10 Stripe	1	100	1000	1000	2000	2000
Mirror (5x mirror)	5	50	500	250-300	1000	1000
RAIDZ1 (9d 1r)	1	90	100	80-90	1800	1600-1800
RAIDZ2 (8d 1r)	1	80	100	70-80	1600	1400-1600
RAIDZ3 (7d 3r)	1	70	100	60-70	1400	1200-1400
dRAID1 (2x RaidZ1 like)	1	80	200-400	150-300	1600-1800	1400-1600
dRAID2 (2x RaidZ2 like)	1	60	200-350	120-250	1600-1800	1200-1400

Keep in mind: AFR (Annualized failure rate) and complexity for each vdev type.

Thank you!

Contact

Deutsches Elektronen-Synchrotron

Tino Reichardt
tino.reichardt@desy.de

www.desy.de

DV Zeuthen