
ZFS

Summary

ZFS is a combined file system and logical volume manager designed and implemented by a team at Sun Microsystems led by Jeff Bonwick and Matthew Ahrens. Its development started in 2001 and it was officially announced in 2004. In 2005 it was integrated into the main trunk of Solaris and released as part of OpenSolaris. Currently, as of January 2015, it is native to Solaris, OpenSolaris, OpenIndiana, illumos, Joyent SmartOS, OmniOS, FreeBSD, Debian GNU/kFreeBSD systems, NetBSD, OSv and supported on Mac OS with MacZFS.

The name "ZFS" originally stood for "Zettabyte File System". Currently it can store up to 256 ZiB (zebibytes).

Installing ZFS on Ubuntu

The ZFS filesystem is available for Ubuntu as either a FUSE module or a native kernel module. The kernel module is provided by default. To install the user-level tools, simply install:

```
sudo apt install zfsutils-linux
```

For all current versions from 16.04 onward.

In addition to be able to have ZFS on root, install:

```
sudo apt install zfs-initramfs
```

See also:

- Ubuntu Kernel Team ZFS Reference Guide

Otherwise:

- Native ZFS for Ubuntu PPA (for 64b systems only)
- Native ZFS for Linux project home.
- ZFS-FUSE project (deprecated).

Rationale

Ubuntu server, and Linux servers in general compete with other Unixes and Microsoft Windows. ZFS is a killer-app for Solaris, as it allows straightforward administration of a pool of disks, while giving intelligent performance and data integrity.

ZFS does away with partitioning, EVMS, LVM, MD, etc. The available disks (of any size) are used to the best of their ability. Compression can be used to increase bandwidth. (q.v. Reiser 4, and cloop?)

ZFS is 128-bit, meaning it is very scalable. (e.g. 16 exabyte limit). Along with its record-setting scalability, it has a number of features that no other file system in production has:

- State-of-the-art *unsurpassed* data reliability (including silent corruption detection and automatic data repair, if available) from end to end, thanks to transparent checksums in parent block pointers
- Elimination of the RAID5 write hole that usually requires battery-powered disk technology or UPS technology to keep RAID arrays consistent in case of power failures
- Online array reconstruction / reassemble, in a fraction of the time usually required to reconstruct an array, thanks to reconstruction dependent on contents rather than on disk topology
- Online check (scrub) and online resilver of faulty, to verify and preserve the validity of on-disk data in the face of silent disk data corruption
- Automatic deduplication and compression of data, selectable per volume or filesystem according to administrator policy

ZFS achieves its impressive performance through a number of techniques:

- Dynamic striping across all devices to maximize throughput
- Copy-on-write design makes most disk writes sequential
- Multiple block sizes, automatically chosen to match workload
- Explicit I/O priority with deadline scheduling
- Globally optimal I/O sorting and aggregation
- Multiple independent prefetch streams with automatic length and stride detection for maximum streaming performance
- Unlimited, instantaneous read/write snapshots

- Parallel, constant-time directory operations

Sun ZFS article (archived on archive.org)

Use cases

- Dani is a sys admin for a medium-size business. She spends 20% of her time planning/changing/administering/repairing the disk arrays of three servers. With ZFS, this time is cut in half.
- Ari has a single disk workstation. She buys a new disk and plugs it in. ZFS automatically adds the new disk space into the pool. Her home directory is mirrored, while her OS and temp space is striped automatically in the background.
- Power user Petra has a four-disk RAID5 workstation. Parity calculations make this a fairly slow set-up because of the number of writes of small files. She upgrades to ZFS and sees performance benefits, because small files are mirrored instead of included in parity calculations.
- Jack has three disks of different sizes. Configuring a sensible partition and RAID set-up is completely trial and error. ZFS abstracts the three disks into one pool of space, and gives the best balance of performance and security. Jack declares that some media directories do not need to be fault-tolerant, and ZFS transparently stripes them across all the disks.
- Manuel has a mirrored disk array. Fluctuations in his computer's PSU are causing silent data corruption errors. ZFS detects the errors and reports back those errors to Manuel, protecting the data already on disk from further corruption.
- Andreas has a disk that has suffered a partial failure. Since he chose a policy of two copies for his personal file system volume, he can recover all of his latest personal data without having to resort to tape backups.
- John has an 18-disk SAS array configured to run mirrored, each nine-disk group containing one RAIDZ2 leg. One of the disks experiences a catastrophic disk motor failure. John deactivates the disk online, replaces it with a fresh one, activates that disk, then uses the ZFS zfs command to replace the faulty disk. ZFS automatically reconstructs the data on that disk with zero downtime and minimal data transfer or performance impact to the array.

ZFS (last edited 2018-02-01 20:19:51 by xennex82 @ localhost[127.0.0.1]:xennex82)