006-ZFS: Zetabyte File System

Notebook:	<Inbox>		
Created:	4/30/2018 8:43 PM	**Updated:**	5/8/2018 11:38 AM
Author:	bdnog8		

Disclaimer:

The scope this topic here is not to discuss about the architecture of the ZFS rather Features, Use Cases and Operational Method.

What is ZFS?

ZFS is a combined file system and logical volume manager designed by Sun Microsystems and now owned by Oracle Corporation. It was 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.

It was described by one analyst as "the only proven Open Source data-validating enterprise file system". This file system also termed as the “world’s safest file system” by some analyst.

ZFS is scalable, and includes extensive protection against data corruption, support for high storage capacities, efficient data compression, deduplication.

ZFS is available for Solaris (and its variants), BSD (and its variants) & Linux (Canonical’s Ubuntu Integrated as native kernel module from version 16.04x)

Features:

- It’s a 128-bit file system that’s capable of managing zettabytes (one billion terabytes) of data.
- Auto Healilng. Everything you do inside of ZFS uses a checksum to ensure file integrity.
- Defense against silent data corruption.
- Dynamicaly expandable* pool based storage management.
- No need to spend time partitioning, formatting, initializing, or doing anything else to your disks – when you need a bigger storage “pool,” just add disks.
- ZFS is capable of many different RAID levels, all while delivering performance that’s comparable to that of hardware RAID controllers. It supports Mirror (RAID1), RAIDZ (RAID5 Like) RAIDZ2 (RAID6 Like), RAIDZ3 (Triple Parity RAID), Hybrid RAID (1+0,51,61,50,60 etc).
- It supports Multi Size Disks.

- It is Copy On Write (COW) File System and support very less costly snapshots & clones. It does not overwrite data in place. ZFS writes a new block to a different spot on the disk and updates the metadata to point to the newly written block, while also retaining older versions of the data.
- File System Replication, Export & Import.
- Disks (full pool) in one system (if hardware of the system fails) can be re-import in a new system without data loss. That is quiet impossible in traditional RAID arrays.
- Supports Quota and it is modifiable on the fly.
- It supports file system level compression.
- No FSCK is require to check disk error.
- Much more less time need to rebuild (re-silver) failed disks. Only actual data is written where as traditional RAID arrays rebuild bit-by-bit every sector of the disk.
- Support read*/write* cache. To increase performance SSD cache can be added.
- Alternative to and much more better than LVM on Linux.

Some Limitation/Controversy:

- ZFS is limited to running on a single server in contrast to distributed or parallel file systems, such as GPFS and Lustre, MooseFS, LizardFS, Ceph/CephFS, Sheepdog, SNFS etc, which can scale out to multiple servers.
- ZFS need more RAM (ECC ram is strongly recommended) and may need more CPU while using deduplication feature.
- In the Linux community, there are various opinions on licensing with respect to the redistribution of the ZFS code and binary kernel modules under a general public license (GPL). But Canonical, which distributes Ubuntu, determined that it is in compliance with the terms of the CDDL and GPL licenses.

Some Commercial Product Using ZFS:

- ORACLE ZFS STORAGE APPLIANCE
- DELPHIX
- NEXENTA
- IXSYSTEMS TRUENAS/FREENAS
- OSNEXUS
- SYNETO
- JOYENT MANTRA

Some Examples Commands:

Installation:

```
apt install zfsutils-linux
```

Pool Creation:

```
zpool create -f vol1 /dev/sda3 ; creating stripe/raid0 pool-volume
zpool create -O ashift=12 -f vol1 mirror /dev/sdc /dev/sdd ; raid0
zpool create -O ashift=12 -f vol1 raidz /dev/sdc /dev/sdd /dev/sde ; raid5
zpool create -O ashift=12 -f vol1 raidz2 /dev/sdc /dev/sdd /dev/sde
/dev/sdf ; raid6

zpool create -O ashift=12 -f vol1 raidz2 /dev/sdc /dev/sdd /dev/sde
/dev/sdf /dev/sdg
; triple parity raid can survie 3-disk failure at a time
```

Pool Manipulation:

```
zpool status  ; see pool status
zpool list    ; see pool list, multiple pool can be created in a single
box

zpool add vol1 raidz /dev/sdf /dev/sdg /dev/sdh ; expand a raidz/raid5
volume

zpool add -f vol1 cache /dev/sdj ; add SSD read cache L2ARC
zpool add -f vol1 cache mirror /dev/sdj /dev/sdk

zpool add -f vol1 log /dev/sdl ; add SSD write cache ZIL
zpool add -f vol1 log mirror /dev/sdl /dev/sdm
```

File System/Dataset Manipulation:

```
zfs list
zfs create vol1/iso
zfs create vol1/kvm
zfs careate vol1/lxc

zfs set compression=on vol1
zfs set atime=off vol1
zfs set autoexpand=on vol1

zfs set quota=10G vol1/lxc
zfs set reservation=10G vol1/lxc
zfs set mountpoint=/mnt/data vol1/iso
```

Snapshot & Snapshot manipulation:

```
zfs create vol1/lxc/container1
zfs snapshot vol1/lxc/container1@version1
zfs snapshot vol1/lxc/container1@`date +%Y%m%d-%H%M%S`

zfs list -t snapshot |grep container1

date_time=`date +%Y%m%d-%H%M%S`
zfs snapshot -r vol1/lxc/container1@$date_time

zfs rollback -r vol1/lxc/container1@version1
zfs destroy -r vol1/lxc/container1@version1
```

Handling Disk Failure

```
zpool replace [poolname] [old drive] [new drive]   ; replace old/damaged
disks
zpool replace vol1 /dev/sdh /dev/sdx

watch zpool status
zpool clear vol1
zpool status
```

Ref-1: https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/
Ref-2: https://www.sotechdesign.com.au/zfs/
Ref-3: https://wp.strahlert.net/wordpress/zfs-2/expanding-zpool/