OpenZFS

= ZFS BASICS & VARIOUS ZFS RAID LEBELS. =

A LITTLE ABOUT ZETABYTE FILE SYSTEM
(ZFS / OpenZFS)

bdNOG11

Disclaimer:

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

What is ZFS?

=0= 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. Matt is the founding member of OpenZFS.

=0= 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.

=0= 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.

=0= ZFS is scalable, and includes extensive protection against data corruption, support for high storage capacities, efficient data compression and deduplication.

=0= 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)

=0= OpenZFS was announced in September 2013 as the truly open source successor to the ZFS project. Opensource community brings together developers from the Illumos (fork of Open

Solaris), FreeBSD, Linux, and OS X platforms, and a wide range of companies that build products on top of OpenZFS.

SOME 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, HDFS, 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, in case of redistribution of the ZFS code as binary kernel modules under a general public license (GPL). But Canonical, which distributes Ubuntu, determined that it is OK! to distribute as binary kernel module.

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
zpool create -o ashift=12 -f vol1 mirror /dev/sdc /dev/sdd
zpool create -o ashift=12 -f vol1 raidz /dev/sdc /dev/sdd /dev/sde
zpool create -o ashift=12 -f vol1 raidz2 /dev/sdc /dev/sdd /dev/sde /dev/sdf
zpool create -o ashift=12 -f vol1 raidz3 /dev/sdc /dev/sdd /dev/sde /dev/sdf /dev/sdg
```

OpenZFS

= ZFS INTENT LOG/ZIL & L2ARC (WRITE & READ CACHE) =

ZFS L2ARC / Read Cache:

In the world of storage, caching can play a big role in improving performance. **OpenZFS** offers some very powerful tools to improve read & write performance.

To improve read performance, ZFS utilizes system memory as an Adaptive Replacement Cache (ARC), which stores your file system's most frequently and recently used data in your system memory. You can then add a Level 2 Adaptive Replacement Cache (L2ARC) to extend the ARC to a dedicated disk (or disks) to dramatically improve read speeds, effectively giving the user **all-flash performance**.

We have to use high-performance SSD/NVME as L2ARC;

Suppose, here **sda15** is our SSD, let's add that SSD as L2ARC,

```
zpool add -f vol1 cache /dev/sda15
```

In case if we want to use two SSD (recommended for reliability/high-availability)

```
zpool add -f vol1 cache mirror /dev/sda15 /dev/sda16
```

```
zpool status
```

```
      cache
        sda15     ONLINE       0     0     0
```

Note: if L2ARC disk fail, pool may freeze for some time but data will be OK!

ZFS ZIL / Write Cache!:

OpenZFS includes something called the ZFS Intent Log (ZIL). The ZIL can be set up on a dedicated disk called a Separate Intent Log (SLOG) similar to the L2ARC, but it is not simply a performance boosting technology.

Many people think of the ZFS Intent Log like they would a write cache. This causes some confusion in understanding how it works and how to best configure it. First of all, the ZIL is more accurately referred to as a "log" whose main purpose is actually for data integrity.

Please read this article http://www.freenas.org/blog/zfs-zil-and-slog-demystified/

We have to use high-performance SSD/NVME as ZIL, Suppose, here **sda16** is our SSD, let's add that SSD as ZIL,

```
zpool add -f vol1 log /dev/sda16
```

In case if we want to use two SSD (recommended for reliability/high-availability)

```
zpool add -f vol1 log mirror /dev/sda16 /dev/sda17
```

```
zpool status
```

```
      logs
        sda16     ONLINE       0     0     0
```

Note: if ZIL disk fail, pool may freeze for some time but data will be OK!

So our final setup is,
zpool status >

```
root@group1-node3:~# zpool status
  pool: vol1
 state: ONLINE
  scan: resilvered 112K in 0h0m with 0 errors on Tue Jul 31 12:27:36 2018
config:

        NAME        STATE     READ WRITE CKSUM
        vol1        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sda10   ONLINE       0     0     0
            sda11   ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            sda14   ONLINE       0     0     0
            sda13   ONLINE       0     0     0
        logs
          sda16     ONLINE       0     0     0
        cache
          sda15     ONLINE       0     0     0
```

As we have to practice LXC on this, so we will remove cache & log device, as here in lab it is not SSD and likely we will get poorer performance.

```
zpool remove vol1 /dev/sda15
zpool remove vol1 /dev/sda16
```

Task: Add two disks (here sda15, sda16) in mirror mode to increase capacity of our existing zfs pool.

Note: It is best-practice for zfs for Linux is to use disk-id instead of /dev/sdx

ll /dev/disk/by-id/ |grep sd |more

```
lrwxrwxrwx 1 root root  11 Jul 31 10:56 ata-ST9500420AS_5VJCLXJQ-part10 -> ../../sda10
lrwxrwxrwx 1 root root  11 Jul 31 10:56 ata-ST9500420AS_5VJCLXJQ-part11 -> ../../sda11
lrwxrwxrwx 1 root root  11 Jul 31 11:41 ata-ST9500420AS_5VJCLXJQ-part12 -> ../../sda12
lrwxrwxrwx 1 root root  11 Jul 31 11:41 ata-ST9500420AS_5VJCLXJQ-part13 -> ../../sda13
lrwxrwxrwx 1 root root  11 Jul 31 12:27 ata-ST9500420AS_5VJCLXJQ-part14 -> ../../sda14
lrwxrwxrwx 1 root root  11 Jul 31 12:46 ata-ST9500420AS_5VJCLXJQ-part15 -> ../../sda15
lrwxrwxrwx 1 root root  11 Jul 31 13:02 ata-ST9500420AS_5VJCLXJQ-part16 -> ../../sda16
```

In that case pools creation command will be

```
zpool create -o ashift=12 -f vol1 mirror /dev/sda10 /dev/sda11 > its equivalent command as follows,
```

```
zpool create -o ashift=12 -f vol1 mirror \
ata-ST9500420AS_5VJCLXJQ-part10 \
ata-ST9500420AS_5VJCLXJQ-part11
```

To see pool io performance,

```
zpool iostat -v
zpool iostat -v 2
```

Useful reading: https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/

ZFS Management | Export, Import, Replication

Practical LAB