The ZFS File System

ZFS is the file system currently owned by Oracle, but available under license to other Unicies. It is an advanced file system with a great advantage to the users.

ZFS is tunable; you can turn flags on and off as you like to support a specific purpose for a mount. By default, ZFS has many flags turned on, resulting in a file system that looks slow compared to other systems you may have used. By turning off the options you do not need, however, you can rapidly speed the system up.

Intro for Linux Sysadmins

ZFS combines the concepts of RAID (limited), LVM2 and DRBD. With added flexibility that your “Logical Volumes” can be tuned to their individual needs, but still dynamically acquire as much space as they need from the Volume Group.

Enabling

FreeBSD does not enable anything by default. To enable ZFS, do the following:

```
cp -av /etc/rc.conf /etc/rc.conf.back
echo 'zfs_enable="YES"' >> /etc/rc.conf
service zfs start
```

Basic ZFS Concepts

The first thing that confused me when I went to ZFS was the concept of a “pool” from which you create datasets. You can think of a pool as a physical disk, and datasets as the partitions on it, though this is a gross simplification. For Linux LVM users, you can think of a pool as a Volume Group and a dataset as a Logical Volume, which is closer, but still a simplification.

A pool is made up of 1 or more disks or partitions (disks are best), similar to RAID arrays. Not quite as powerful as Linux's mdadm, but still good. Think hardware RAID, or mdadm about 20 years ago. Creating a pool is simple:

```
zpool create poolname redundancy_type device device device
```

poolname is anything you want to call it (default is zpool, I think), and redundancy type is one of the limited ones that ZFS supports, mainly *mirror* (RAID-1) and *raidz2* (RAID-5), though by leaving redundancy_type out, you can use a single disk.

View your pool after you have done the above with:

```
zpool list
```

One nice thing about the zpool (and zfs command below) is that you can add -v to get more information. Adding more -v's gives additional information. So, for example:

```
# basic information on the pools
zpool list
# more information is displayed
zpool list -v
# even more, if it was appropriate (in this case it is not)
zpool list -vv
```

Another point of confusion is the pool will show itself to be “mounted” from commands like *df* as its own directory, like */zpool/*. Note that this is NOT a mounted system.

Enough of that. A pool can be broken into individual *datasets*. Again, you can think of them as a partition, or a Logical Volume, in that you can tune them. Not just things like no atime or no dev's, but setting up a blocksize, quota, buffering, logging. What you could normally do on other file systems, but you do this while still having full access to the space of the entire pool. This is what confused me.

Creating a dataset is again, quite easy.

```
zfs create poolname/datasetname
```

The command above just creates the pool with the defaults. You can set the options when you create it, or after. So, for example:

```
zfs create -o mountpoint=/opt/ds1 -o compress=lz4 poolname/datasetname
```

creates datasetname as a compressed file system (using lz4) which is automounted on /opt/ds1. The following does the same:

```
zfs create poolname/datasetname
zfs set mountpoint=/opt/ds1 poolname/datasetname
zfs set compress=lz4 poolname/datasetname
```

To see all of the options (flags) available for a dataset, after you've created it, issue the command:

```
zfs get all poolname/datasetname
```

all means get all options; you can give it the name if you like, such as

```
zfs get mountpoint poolname/datasetname
```

The nice thing about it is you can modify an existing dataset, on the fly. Thus, if you do not want a particular mount point, simply set it to a different one.

```
zfs set mountpoint=/opt/ds2 poolname/datasetname
```

The main power you have at this point is that you still have access to all of the original disk space

(unless you've set a quota). So, you could set /home with one set of parameters, /tmp with another, /var/log with still another, but you don't have to worry about a partition being too small. Each can have their own block sizes, buffering, etc…

ZFS Volumes

To quote from the FreeBSD ZFS Administrators Manual:

A volume is a special type of dataset. Rather than being mounted as a file system, it is exposed as a block device under /dev/zvol/poolname/dataset. This allows the volume to be used for other file systems, to back the disks of a virtual machine, or to be exported using protocols like iSCSI or HAST.

zfs volumes are created using the -V parameter, and statically creating a size on creation. The basic syntax is as follows (volblocksize is set to the default, 8k, and volmode=dev gives faster access when used over iSCSI). Using a volmode=full will allow greater functionality from within the machine containing the ZFS file system (ie, mounting it there).

```
zfs create -V 10G -o volblocksize=8K,volmode=dev storage/virtual_2
```

which will create a 10 Gig container named virtual_2. This 10G is allocated out of the zpool storage/

NOTE: you can access volumes (for exporting via iSCSI) under the directory /dev/zvol/zpoolname/zvolname

The zvol can be grown or reduced by setting the volsize property. Obviously be careful of this if you grow or reduce something with data and a file system on it. Several other setting can be changed during or after creation, though some will only work on *new* data after set.

```
zfs set volsize=15G storage/virtual_2
zfs set checksum=off storage/virtual_2
zfs set compression=off storage/virtual_2
zfs set readonly=on storage/virtual_2
```

You can take a snapshot of a volume, and the space for the snapshot is allocated out of the zpool it is in.

Specialized Tuning

These are just some recipes I am storing so I remember how to do them. As I find more, I'll add them here. In each case, I'm creating the dataset with all the options at the start.

Set ZFS for MySQL Store

One area that definitely needs tuning is when you create a mount specifically for a database. The following assumes MariaDB/MySQL on FreeBSD, but it equally applicable to PostGreSQL and Oracle

(though some of the parameters and locations should likely change).

NOTE: this should be done on a new install. It is assumed mySQL has never run on this machine, so the data directories are not populated (on FreeBSD, the first run of mySQL creates the data files. If you have run it, back up any data, then

```
# WARNING,WARNING,WARNING
# this will delete any existing databases
rm -fR /var/db/mysql/*
```

When you next run mySQL, the basic databases and tables will be created.

Verify you have empty directories, then execute the following commands (replacing pool/ with whatever you created your zpool as, of course). You will also want to modify your my.cnf (/usr/local/etc/my.cnf, probably doesn't exist, so copy one from /usr/share/doc/mysql).

```
# create our datasets
zfs create pool/db/logs
zfs create pool/db/innodb
zfs create pool/db/myisam

# set global paraemters
zfs set zfs:zfs_nocacheflush = 1
zfs set setuid=off pool/db
zfs set primarycache=metadata pool/db
zfs set atime=off pool/db
zfs set sync=disabled pool/db
zfs set compression=off pool/db
zfs set logbias=throughput pool/db

# now, set individual parameters (blocksizes)
zfs set recordsize=16k pool/db/innodb
zfs set recordsize=128k pool/db/logs
zfs set recordsize=8k pool/db/myisam

# create all directories, set the mount points, then set ownership
mkdir /var/log/mysql
zfs set mountpoint=/var/log/mysql pool/db/logs
chown mysql:mysql /var/log/mysql

mkdir /var/db/mysql
zfs set mountpoint=/var/db/mysql pool/db/myisam
chown mysql:mysql /var/db/mysql

mkdir /var/db/mysql-innodb
zfs set mountpoint=/var/db/mysql-innodb pool/db/innodb
chown mysql:mysql /var/db/mysal-innodb
```

In /usr/local/etc/my.cnf (FreeBSD) or /etc[/mysql]/my.cnf (Linux)

```
[mysqld]
```

```
# move binlogs to their own directory, might as well use /var/log
log_bin = /var/log/mysql/mysql-bin.log
# set innodb to be located here, and give it a couple of data files
innodb_data_home_dir = /var/db/mysql-innodb/
innodb_data_file_path=ibdata1:2G;ibdata2:10M:autoextend
innodb_flush_log_at_trx_commit=2
skip-innodb_doublewrite
# be careful with this if you load HUGE files, it could get too large
# read next section on how to set up /tmp
tmpdir=/tmp
```

Set ZFS for temp file system

```
zfs create -o sync=disabled    \
           -o devices=off      \
           -o setuid=off       \
           -o mountpoint=/tmp  \
           <pool>/tmp
```

- sync - do not support application sync command
- devices - do not allow device creation
- setuid - allow us to set perms on directory

Bibliography

- https://wiki.archlinux.org/index.php/ZFS
- https://www.freebsd.org/doc/handbook/zfs-zpool.html
- https://www.freebsd.org/doc/handbook/zfs-zfs.html
- https://dev.mysql.com/doc/refman/5.5/en/optimizing-innodb-diskio.html
- https://wiki.freebsd.org/ZFSTuningGuide

From:
https://wiki.linuxservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://wiki.linuxservertech.com/unix/freebsd/zfs

Last update: **2025/04/11 22:24**