

ZFS on FUSE

By Roderick W. Smith

Although its features and terminology may seem strange if you're used to more traditional Linux filesystems, ZFS offers a great deal of flexibility.

In recent years, filesystems have undergone dramatic changes. In the Linux world, ext2fs has long been the standard filesystem, but its journaling successor, ext3fs, has become prevalent, along with other journaling filesystems. Most recently, a new upstart, the Zettabyte File System (ZFS), has been gaining attention.

As described shortly, ZFS boasts many revolutionary features, although it's also got some serious problems from a Linux perspective. Despite these problems, ZFS is worth investigating — if nothing else, as a hint of future filesystem developments generally.

ZFS was released under Sun's Common Development and Distribution License (CDDL), which is incompatible with the GNU Public License (GPL) used by Linux. For this reason, ZFS is only available for Linux in the form of a Filesystem in Userspace (FUSE) module. This means that, despite its promise, ZFS is not a good choice to use as the only filesystem on a Linux system, although you can use it for storing user files and other non-system files. ZFS for Linux is still fairly immature, so I recommend you treat it as experimental.

If its features are extremely important to you, you might want to consider looking at OpenSolaris, which has more mature ZFS support. If you'd like to get an advance look at ZFS and try it out on some non-mission-critical systems, though, using ZFS on FUSE is a perfectly reasonable alternative.

Why Use ZFS?

So, just what are these advanced features of ZFS? Its developers re-examined filesystem design from the ground up. The result, according to the ZFS Web page, is that its developers have “blown away 20 years of obsolete assumptions, eliminated complexity at the source, and created a storage system that's actually a pleasure to use.” Examples of its advanced features include:

- Pooled storage — The filesystem enables storage space from multiple disks to be “pooled” and allocated to specific mount points on an as-needed basis. This feature effectively combines volume-resizing and Logical Volume Management (LVM) features, but at the filesystem level. With ZFS, you don't need to worry about how to size your partitions, since the filesystem adjusts itself automatically!
- RAID-Z — This feature is similar to Redundant Array of Independent Disks (RAID) level 5 support, but with less overhead.
- Always-consistent disk space — The disk space is kept consistent at all times through copy-on-write operations. Thus, there's no need to perform disk checks after a system crash or power outage.
- Disk scrubbing — This feature is similar to the Error Correcting Code (ECC) feature of certain computer memory modules; it permits the computer to detect and correct on-disk errors.
- Snapshots and clones — A snapshot is a read-only copy of a filesystem, and a clone is a read/write copy of a filesystem. You can use snapshots and clones to preserve a fixed version of a filesystem or to make backups of a filesystem.
- Built-in compression — ZFS supports compression at the filesystem level, which is particularly handy if you're low on disk space or if you store highly compressible data.

ZFS on FUSE

By Roderick W. Smith

These features make ZFS an appealing filesystem for many purposes. The pooled storage model alone should be enough to pique the interest of anybody who's managed Linux systems using old-style partitions!

The main drawback to ZFS on Linux include its early status (as I write, it's currently at version 0.4.0 beta 1) and the fact that it must be implemented via a FUSE module. These factors mean that there are certain things that ZFS on Linux can't do. Most notably, ZFS for FUSE doesn't yet support Access Control Lists (ACLs), exporting via the Network File System (NFS), or storing swap space on a ZFS volume.

Check the STATUS file in the ZFS distribution to see a complete list of unsupported features. ZFS for FUSE has extensive memory requirements; it chews up about 128MB of RAM, and its developers recommend that it be used only on systems with 1GB or more of RAM. Because ZFS for Linux is only available via FUSE, it's not a good choice for your root filesystem. You should perform stability and performance tests on your own hardware before entrusting critical data to ZFS.

FUSE Basics

Because ZFS for Linux is a FUSE module, you must first get FUSE up and running before you can use ZFS. For more on FUSE you can check out my earlier article (Lighting the FUSE) or the main FUSE Web page, for details on using FUSE. A quick introduction should be enough to get you going, though.

The main FUSE package, available from its Web site, includes user-mode tools and Linux kernel modules. Chances are your distribution includes a ready-made FUSE package (Fedora and Gentoo both provide packages called fuse, and Ubuntu provides one called fuse-utils, for instance.) Use synaptic, yum, emerge, or whatever package-management tools your distribution provides to locate and install this package. If you can't locate a distribution-specific package, you can download the source code from the main FUSE Web site.

FUSE looks for kernel support while it's building, and if it doesn't find that support, FUSE will build it. If you prefer, though, you can build the kernel modules as part of a standard Linux kernel compilation — FUSE is available as a regular filesystem option. Once installed, the main FUSE package provides support libraries, documentation, and user utilities. The most important of these user tools is fusermount, which is used to mount FUSE-based filesystems.

Obtaining and Installing ZFS for FUSE

Few distributions provide ready-made ZFS for FUSE packages (Gentoo is an exception). If your distribution provides such a package, it will likely be the easiest way to add ZFS support to your system; however, you should check the version to be sure it's not too far behind the latest version available on the ZFS for FUSE Web site.

If you must install ZFS from source code, the steps are fairly typical, with a notable exception:

1. Download the tarball (zfs-fuse-0.4.0_beta1.tar.bz2 is the latest as I write).
2. In a suitable directory, uncompress the tarball by typing `tar xvfj zfs-fuse-0.4.0_beta1.tar.bz2`. The result is a subdirectory containing the source code, documentation, and support files.
3. Read the README, INSTALL, and STATUS files to get up to speed on the status of the software and so you'll be aware of any important deviations from what I describe here.

ZFS on FUSE

By Roderick W. Smith

4. Change into the src subdirectory and type `scons`. The `scons` program is similar to the more commonly-used `make` program; it directs the compilation process to compile the ZFS code.
5. Type `scons install` as root to install the ZFS files.

As you might guess by steps four and five, ZFS for FUSE relies on `scons`. This program is likely to be available as a ready-made package for your system, but if not, you can obtain its source code from `scons.org`. ZFS for FUSE also requires Linux kernel version 2.6.x (2.6.15 or later being recommended), FUSE version 2.5.x or later (although Gentoo's 2.6.0rc1 release doesn't work), and `glibc` 2.3.3 or later. Version 0.4.0 beta 1 works only with x86 and AMD64 CPUs.

Using ZFS

In order to function, ZFS requires the `zfs-fuse` program to be running (launched as root). You can launch this program by using the `run.sh` script in the ZFS for FUSE `src/zfs-fuse` directory. Gentoo provides a suitable startup script (`/etc/init.d/zfs-fuse`) as part of its SysV init script inventory. If you don't have such a script, and if you intend to use ZFS on a regular basis, you'll probably want to modify your local startup scripts to launch `run.sh`, or cut-and-paste its contents into an existing startup script. Note that `zfs-fuse` isn't a true daemon, so you'll want to launch it in the background by appending an ampersand (&) to its command line in your local startup script.

Before proceeding further, you should understand the distinction between pools and data sets. A pool is a named collection of one or more partitions or disks. Each pool may contain one or more data sets, each of which can be mounted at a different mount point. When you create a ZFS pool, it may be immediately accessed as a data set, and you can create and delete additional data sets within the pool as you see fit. You'll use two commands, `zpool` and `zfs`, to manipulate pools and data sets. With the `zfs-fuse` program running, your first step is to prepare one or more partitions to be part of a ZFS pool. You can do this with the `zpool` command:

```
zpool create mypool /dev/sda1
```

This example creates a ZFS pool, and associated initial data set, on the `/dev/sda1` device and mounts the data set at `/mypool`. You should of course change the pool name and source device for your system. If you want to change the mount point, you can use the `zfs` utility and its `set` option:

```
zfs set mountpoint=/usr/local mypool
```

This command changes the mount point from `/mypool` to `/usr/local`. In the process, the data set may be unmounted. You can remount it using the `zfs` utility's `mount` option, in one of two ways:

```
zfs mount -a
zfs mount mypool
```

The `-a` option mounts all the defined data sets, whereas the second mounts only the specified data set. You can include one of these commands in a local startup script if you want your ZFS data sets to be accessible whenever you boot the system. If you want to unmount a data set, you do so with the `zfs unmount` command:

```
zfs unmount -a
zfs unmount mypool
```

Note that the subcommand name is `unmount`, with two `n`s, unlike the Linux `umount` command, which has just one `n`. As you might guess, these two commands unmount all ZFS data sets or just the specified data set, respectively. Note that you should generally use the `zfs` tool to manipulate ZFS

ZFS on FUSE

By Roderick W. Smith

data sets rather than the standard Linux mount and umount commands. ZFS remembers the mount points, so you don't need to specify them on the command line or in /etc/fstab.

If you want to use a pool to store data under two directories (say, /opt and /usr/local), you can use zfs to create a new data set within the pool:

```
zfs create mypool/opt
zfs set mountpoint=/opt mypool/opt
zfs mount -a
```

At this point, the /dev/sda1 device you dedicated to the pool acts much like two partitions, one mounted at /usr/local (the data set associated with the original pool) and the other at /opt (the new data set); however, disk space will be allocated dynamically, so that if you end up storing more data in /opt and less in /usr/local than you'd anticipated, you won't have any problems.

If you run out of disk space in your initial pool, you can expand it using zpool's add command (provided you've got a free disk or partition):

```
zpool add mypool /dev/sdb1
```

This example adds /dev/sdb1 to the existing mypool pool. Be aware, however, that removing devices from a pool isn't as easy as adding them; thus, you shouldn't add devices to a pool just to test this feature.

Before shutting down the system, you should issue the zpool export command. You should also issue this same command before moving the volume to a non-Linux system (say, if you use ZFS on a removable disk). This command cleans up the system and makes your ZFS pools inaccessible until you use the zpool import mypool command, where mypool is the name of the pool you want to use.

The ZFS utilities then make this pool available and mount it. If you don't remember a pool name, typing zpool import causes the ZFS utilities to scan the disks for available pools.

Advanced ZFS Tricks

Unfortunately, man pages relevant to ZFS have not yet been ported to Linux, as of ZFS for FUSE 0.4.0 beta 1. You can, however, learn something of the relevant commands by typing zpool or zfs with no options; the result is a list of subcommands and the options they take. Documentation on Sun's main ZFS site can also help you on your way. Some highlights of advanced features of ZFS include:

- Compression — You can enable compression by typing zfs set compression=on mypool (substituting your data set name, of course).
- Quotas — You can give quotas to particular data sets, as in zfs set quota=10G mypool/opt to give the mypool/opt data set a 10GB quota. Disk usage won't be permitted to exceed the specified value for this data set.
- Reservations — You can reserve at least a given amount of space for a specific data set, as in zfs set reservation=5G mypool/opt. This command guarantees that the mypool/opt data set will be able to store at least 5GB of data.
- Disk checks — Type zpool scrub mypool (substituting your pool name) to check the data integrity on the pool. This is equivalent to the fsck command.

ZFS on FUSE

By Roderick W. Smith

- Replacing disks — If you discover that a disk is flaky or you otherwise want to replace it, you can use the `replace` subcommand to `zpool`, as in `zpool replace mypool /dev/sda1 /dev/sdc1`. This example replaces `/dev/sda1` with `/dev/sdc1`. This command can take a while to execute, since a lot of data may need to be copied.
- Creating snapshots — To create a snapshot (a read-only copy) of a data set, use the `zfs snapshot` command, as in `zfs snapshot mypool@backup`. This example creates a snapshot of `mypool` called `backup`. Note that the snapshot is stored in the same pool as the original.
- Rolling back a snapshot — You can use the `rollback` option to `zfs` to restore a snapshot, as in `zfs rollback mypool@backup` to restore the filesystem to the state it was in when the backup snapshot was taken.
- Destroying data sets and pools — The `destroy` option to `zfs` destroys data sets, while the same option to `zpool` destroys an entire pool, as in `zfs destroy mypool/opt` or `zpool destroy mypool`.

These examples don't exhaust all you can do with ZFS. Although its features and terminology may seem strange at first if you're used to more traditional Linux filesystems, ZFS offers a great deal of flexibility. You can learn more by experimenting with this filesystem (on a non-critical system, of course!) or by reading the documentation on the Web sites I've already referenced.