Native ZFS support on GNU/Linux

Finnbarr P. Murphy

(fpm@fpmurphy.com)

ZFS (Zettabyte File System) is a 128-bit advanced file system and volume manager developed by Sun Microsystems (now part of Oracle) as open-source software licensed under the Common Development and Distribution License (CDDL). When first introduced in 2004, ZFS was an entirely new approach to managing disk storage space and protecting the integrity of data. All metadata is stored dynamically and file systems do not need to preallocate inodes or other space when created. Other advantages include scalability, provable data integrity, atomic updates to ensure data consistency, instantaneous snapshots and clones, optional data compression and block-level de-duplication.

The four major components of ZFS are the Storage Pool Allocator (SPA), the Data Management Unit (DMU), the ZFS Attribute Processor (ZAP) and the ZFS Posix Layer (ZPL). The SPA handles all block allocation and freeing and also IO. It abstracts devices into *vdevs* and provides virtually addressed blocks to the DMU which transforms the virtually addressed blocks into transactional object interfaces for the ZPL. The DMU is responsible for presenting a transactional object model and maintaining the data consistency using Copy on Write (COW) semantics and atomic operations. Consumers interact with the DMU via *objsets*, objects, and transactions. An *objset* is a collection of objects, where each object is an arbitrary piece of storage from the SPA. ZPL implements a POSIX compliant system on top of the DMU. ZAP also sits on top of the DMU and is used to create arbitrary (name, object) associations within an *objset*.

Here is a diagram shamelessly borrowed from the OpenSolaris ZFS Source Tour webpage which shows the various components of ZFS:

If you want to learn more about the internals of ZFS, two good starting points are the OpenSolaris ZFS Tour and Solaris Internals -ZFS Performance.

So why is ZFS not currently available in mainstream GNU/Linux distrubutions? The answer is that the Free Software Foundation (FSF) takes the position that CDDL is incompatible with the GNU General Public License (GPL). This license conflict has blocked the inclusion of ZFS in the Linux kernel but not in other platforms such as FreeBSD and NetBSD. In addition, BTRFS appeared, gained significant mind share in the GNU/Linux community, and was accepted into the Linux kernel in spite of the fact that it is still under heavy development.

A port of ZFS to GNU/Linux using FUSE (Filesystem in Userspace) was done as a 2006 Google Summer of Code project. Under FUSE, ZFS runs as a userspace filesystem and thus is not considered a derived work of the kernel. However ZFS-FUSE has never proven to be popular. The current version of ZFS-FUSE is functionally equivalent to the zpool v23 release in Solaris 10.

Recently a native port of ZFS to GNU/Linux was done by a small team at the Lawrence Livermore National Laboratory (LLNL) to augment the Lustre networking filesystem with ZFS support. The port currently includes a fully functional and stable SPA, DMU, and ZVOL but does not have ZPL. A Pune India, company called KQ Infotech Pvt. Ltd has taken the LLNL code, added a ZPL and enhanced it so that it is functionally equivalent to the zpool v28 release in Solaris 10 9/10. This port entered a closed beta last year with GA on January 14th, 2011.

The binary packages can be downloaded from kqstor.com. A number of GNU/Linux distributions are currently supported:

Package	Kernel Version	Download
RHEL 6 (x86_64)	2.6.32-71.el6	KQ Stor binaries for RHEL6
Fedora 14 (x86_64)	2.6.35.10-74	KQ Stor binaries for Fedora 14
Ubuntu 10.10 (x86_64)	2.6.35-22-server	KQ Stor binaries for Ubuntu 10.10 Server.
Ubuntu 10.04 (x86_64)	2.6.35-22-server	KQ Stor binaries for Ubuntu 10.04 Server.

Note that only 64-bit platforms are supported,

I decided to download and test their 64-bit Fedora 14 package. After filling in a detailed user registration form asking me all sorts of questions including whether I intended to use ZFS port to Linux for commercial purposes or whether I wished to become a KQ technology partner , I was granted access to the download area. Interestingly, their website uses Microsoft IIS and Active Server Pages (ASP) rather than Gnu/Linux!

Here is a list of the files that are in the *zfs-fedora-2.6.35.10-74.fc14.x86_64.tar.gz* tarball that I downloaded:

```
./zfs-fedora-2.6.35.10-74.fc14.x86_64
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl/spl-0.5.2-2.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl/spl-modules-devel-0.5.2-2_2.6.35.10_74.fc14.x86_
64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl/spl-0.5.2-2.src.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl/spl-modules-0.5.2-2.src.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/spl/spl-modules-0.5.2-2_2.6.35.10_74.fc14.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/lzfs
./zfs-fedora-2.6.35.10-74.fc14.x86_64/lzfs/lzfs-1.0-2_2.6.35.10_74.fc14.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/lzfs/lzfs-1.0-2.src.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-0.5.1-2.src.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-test-0.5.1-2.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-modules-0.5.1-2.src.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-modules-0.5.1-2_2.6.35.10_74.fc14.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-0.5.1-2.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-devel-0.5.1-2.x86_64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/zfs/zfs-modules-devel-0.5.1-2_2.6.35.10_74.fc14.x86_
64.rpm
./zfs-fedora-2.6.35.10-74.fc14.x86_64/user-guide.pdf
./zfs-fedora-2.6.35.10-74.fc14.x86_64.tar.gz
```

I will not bother showing you what is in the source or development RPMs but here is a list of the files included in the binary RPMs:

```
[zfs]$ rpm -qlp zfs-0.5.1-2.x86_64.rpm
/etc/udev
/etc/udev/rules.d
/etc/udev/rules.d/60-zpool.rules
/etc/zfs
/etc/zfs/zdev.conf
/etc/zfs/zdev.conf.dragon.example
/etc/zfs/zdev.conf.supermicro.example
/etc/zfs/zdev.conf.x4550.example
/usr/bin/zpool_id
/usr/bin/zpool_layout
```



```
/usr/lib64/libavl.a
/usr/lib64/libavl.la
/usr/lib64/libavl.so
/usr/lib64/libavl.so.0
/usr/lib64/libavl.so.0.0.0
/usr/lib64/libefi.a
/usr/lib64/libefi.la
/usr/lib64/libefi.so
/usr/lib64/libefi.so.0
/usr/lib64/libefi.so.0.0.0
/usr/lib64/libnvpair.a
/usr/lib64/libnvpair.la
/usr/lib64/libnvpair.so
/usr/lib64/libnvpair.so.0
/usr/lib64/libnvpair.so.0.0.0
/usr/lib64/libspl.a
/usr/lib64/libspl.la
/usr/lib64/libspl.so
/usr/lib64/libspl.so.0
/usr/lib64/libspl.so.0.0.0
/usr/lib64/libunicode.a
/usr/lib64/libunicode.la
/usr/lib64/libunicode.so
/usr/lib64/libunicode.so.0
/usr/lib64/libunicode.so.0.0.0
/usr/lib64/libuutil.a
/usr/lib64/libuutil.la
/usr/lib64/libuutil.so
/usr/lib64/libuutil.so.0
/usr/lib64/libuutil.so.0.0.0
/usr/lib64/libzfs.a
/usr/lib64/libzfs.la
/usr/lib64/libzfs.so
/usr/lib64/libzfs.so.0
/usr/lib64/libzfs.so.0.0.0
/usr/lib64/libzpool.a
/usr/lib64/libzpool.la
/usr/lib64/libzpool.so
/usr/lib64/libzpool.so.0
/usr/lib64/libzpool.so.0.0.0
/usr/sbin/zdb
/usr/sbin/zfs
/usr/sbin/zinject
/usr/sbin/zpios
/usr/sbin/zpool
/usr/sbin/ztest
/usr/share/doc/zfs-0.5.1
/usr/share/doc/zfs-0.5.1/AUTHORS
/usr/share/doc/zfs-0.5.1/COPYING
/usr/share/doc/zfs-0.5.1/COPYRIGHT
/usr/share/doc/zfs-0.5.1/ChangeLog
/usr/share/doc/zfs-0.5.1/DISCLAIMER
/usr/share/doc/zfs-0.5.1/META
/usr/share/doc/zfs-0.5.1/OPENSOLARIS.LICENSE
/usr/share/doc/zfs-0.5.1/README.markdown
/usr/share/doc/zfs-0.5.1/ZFS.RELEASE
/usr/share/man/man8/zdb.8.gz
/usr/share/man/man8/zfs.8.gz
/usr/share/man/man8/zpool.8.gz

[zfs]$ rpm -qlp zfs-modules-0.5.1-2_2.6.35.10_74.fc14.x86_64.rpm
/lib/modules/2.6.35.10-74.fc14.x86_64
/lib/modules/2.6.35.10-74.fc14.x86_64/addon
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/avl
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/avl/zavl.ko
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/nvpair
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/nvpair/znvpair.ko
```



```
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/unicode
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/unicode/zunicode.ko
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zcommon
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zcommon/zcommon.ko
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zfs
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zfs/zfs.ko
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zpios
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/zfs/zpios/zpios.ko

[lzfs]$ rpm -qlp lzfs-1.0-2_2.6.35.10_74.fc14.x86_64.rpm
/etc/init.d/zfsload
/etc/zfs/zfs_config
/etc/zfs/zfs_serial
/lib/modules/2.6.35.10-74.fc14.x86_64
/lib/modules/2.6.35.10-74.fc14.x86_64/addon
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/lzfs
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/lzfs/lzfs.ko
/usr/sbin/SystemReport.sh
/usr/sbin/zfs_manage.py
/usr/share/doc/lzfs-1.0
/usr/share/doc/lzfs-1.0/COPYING
/usr/share/doc/lzfs-1.0/ChangeLog
/usr/share/doc/lzfs-1.0/META
/usr/src/lzfs-1.0
/usr/src/lzfs-1.0/2.6.35.10-74.fc14.x86_64
/usr/src/lzfs-1.0/2.6.35.10-74.fc14.x86_64/lzfs_config.h
/usr/src/lzfs-1.0/2.6.35.10-74.fc14.x86_64/module
/usr/src/lzfs-1.0/2.6.35.10-74.fc14.x86_64/module/Module.symvers

[spl]$ rpm -qlp spl-0.5.2-2.x86_64.rpm
/usr/sbin/spl
/usr/sbin/splat
/usr/share/doc/spl-0.5.2
/usr/share/doc/spl-0.5.2/AUTHORS
/usr/share/doc/spl-0.5.2/COPYING
/usr/share/doc/spl-0.5.2/ChangeLog
/usr/share/doc/spl-0.5.2/DISCLAIMER
/usr/share/doc/spl-0.5.2/INSTALL
/usr/share/doc/spl-0.5.2/META

[spl]$ rpm -qlp spl-modules-0.5.2-2_2.6.35.10_74.fc14.x86_64.rpm
/lib/modules/2.6.35.10-74.fc14.x86_64
/lib/modules/2.6.35.10-74.fc14.x86_64/addon
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/spl
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/spl/spl
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/spl/spl/spl.ko
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/spl/splat
/lib/modules/2.6.35.10-74.fc14.x86_64/addon/spl/splat/splat.ko
[spl]$
```


The source code is available at *https://github.com/zfs-linux/zfs*. The majority of the source code appears to come from OpenSolaris. This includes the core ZFS code, libavl, *libnvpair*, *libefi*, *libunicode*, and *libutil*.

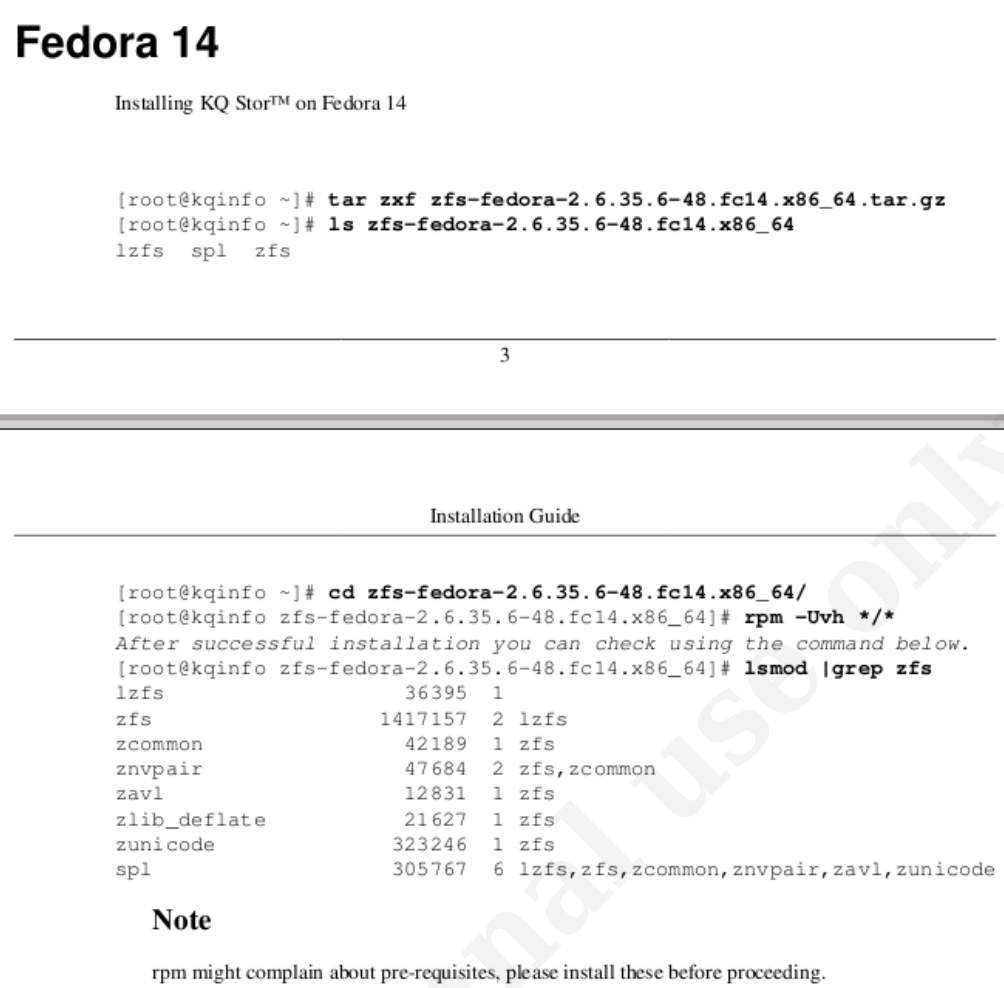
A minor problem that I encountered is that */usr/lib64/libefi.a* conflicts with the */usr/lib/libefi.a* in the existing *gnu-efi-3.0e-11.fc14.x86_64.rpm* (Development Libraries and Headers for EFI) package. However, unless you are doing Extensive Firmware Interface (EFI) development work, you should not encounter this problem. This issue just needs to be mentioned in the Installation Guide or Release Notes.

Another disconcerting fact is that the ZFS-related shared libraries are missing actual version numbers.


```
# cd /usr/lib64
[lib64]# ls -al libuutil.so* libunicode.so* libuutil.so* libspl.so* libnvpair.so* libefi.s
o* libavl.so* libzfs.so* libzpool.so*
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libavl.so -> libavl.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libavl.so.0 -> libavl.so.0.0.0
-rwxr-xr-x 1 root root   28437 Jan 21 07:27 libavl.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libefi.so -> libefi.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libefi.so.0 -> libefi.so.0.0.0
-rwxr-xr-x 1 root root   48667 Jan 21 07:27 libefi.so.0.0.0
lrwxrwxrwx 1 root root      18 Jan 23 00:45 libnvpair.so -> libnvpair.so.0.0.0
lrwxrwxrwx 1 root root      18 Jan 23 00:45 libnvpair.so.0 -> libnvpair.so.0.0.0
-rwxr-xr-x 1 root root  196101 Jan 21 07:27 libnvpair.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libspl.so -> libspl.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libspl.so.0 -> libspl.so.0.0.0
-rwxr-xr-x 1 root root   73002 Jan 21 07:27 libspl.so.0.0.0
lrwxrwxrwx 1 root root      19 Jan 23 00:45 libunicode.so -> libunicode.so.0.0.0
lrwxrwxrwx 1 root root      19 Jan 23 00:45 libunicode.so.0 -> libunicode.so.0.0.0
-rwxr-xr-x 1 root root  376856 Jan 21 07:27 libunicode.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libuutil.so -> libuutil.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libuutil.so -> libuutil.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libuutil.so.0 -> libuutil.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libuutil.so.0 -> libuutil.so.0.0.0
-rwxr-xr-x 1 root root  111416 Jan 21 07:27 libuutil.so.0.0.0
-rwxr-xr-x 1 root root  111416 Jan 21 07:27 libuutil.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libzfs.so -> libzfs.so.0.0.0
lrwxrwxrwx 1 root root      15 Jan 23 00:45 libzfs.so.0 -> libzfs.so.0.0.0
-rwxr-xr-x 1 root root  689431 Jan 21 07:27 libzfs.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libzpool.so -> libzpool.so.0.0.0
lrwxrwxrwx 1 root root      17 Jan 23 00:45 libzpool.so.0 -> libzpool.so.0.0.0
-rwxr-xr-x 1 root root 3061347 Jan 21 07:27 libzpool.so.0.0.0
```

This should be corrected in a future release.

A user guide is included but it is cryptic to the point of being useless as far as installation by an inexperienced user is concerned.



Fedora 14

Installing KQ Stor™ on Fedora 14

```
[root@kqinfo ~]# tar zxf zfs-fedora-2.6.35.6-48.fc14.x86_64.tar.gz
[root@kqinfo ~]# ls zfs-fedora-2.6.35.6-48.fc14.x86_64
lzfs  spl  zfs
```

3

Installation Guide

```
[root@kqinfo ~]# cd zfs-fedora-2.6.35.6-48.fc14.x86_64/
[root@kqinfo zfs-fedora-2.6.35.6-48.fc14.x86_64]# rpm -Uvh */*
After successful installation you can check using the command below.
[root@kqinfo zfs-fedora-2.6.35.6-48.fc14.x86_64]# lsmod |grep zfs
lzfs                   36395  1
zfs                  1417157  2 lzfs
zcommon                42189  1 zfs
znvpair                47684  2 zfs,zcommon
zavl                   12831  1 zfs
zlib_deflate           21627  1 zfs
zunicode              323246  1 zfs
spl                   305767  6 lzfs,zfs,zcommon,znvpair,zavl,zunicode
```

Note

rpm might complain about pre-requisites, please install these before proceeding.

I found it best to just select the RPMs that I wanted and install them individually, resolving missing dependencies along the way. For me, the *lzfs* package was the only package that had missing dependencies.

```
[lzfs]# rpm -Uvh lzfs-1.0-2_2.6.35.10_74.fc14.x86_64.rpm
error: Failed dependencies:
        SOAPpy is needed by lzfs-1.0-2_2.6.35.10_74.fc14.x86_64
        python-dmidecode is needed by lzfs-1.0-2_2.6.35.10_74.fc14.x86_64
```

As the system that I am testing ZFS on is a fully loaded development system, I suspect the regular user will encounter more unresolved dependencies than this.

The *lzfs* package should be the last to be installed and when successfully installed the ZFS modules are loaded into the kernel.

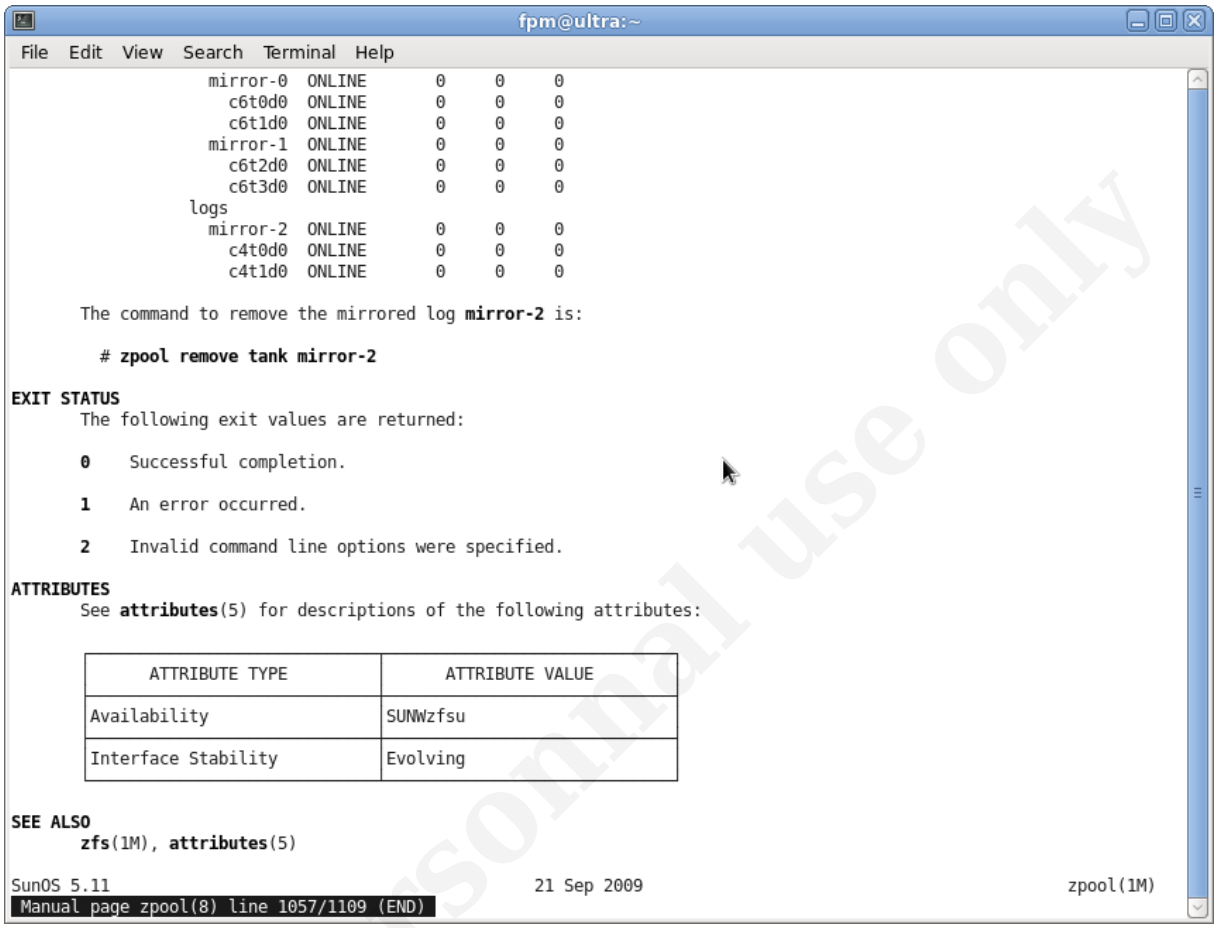
```
[lzfs]# rpm -Uvh lzfs-1.0-2_2.6.35.10_74.fc14.x86_64.rpm
Preparing...                ########################################### [100%]
   1:lzfs                   ########################################### [100%]
loading zfs modules                                        [  OK  ]

[lzfs]# lsmod | grep zfs
lzfs                   32790  0
zfs                   821704  1 lzfs
zcommon                35628  1 zfs
znvpair                38001  2 zfs,zcommon
zavl                    5688  1 zfs
```



```
zlib_deflate           18639  1 zfs
zunicode              319101  1 zfs
spl                   105294  6 lzfs,zfs,zcommon,znvpair,zavl,zunicode
```

Interestingly, the three ZFS man pages included are straight from Solaris 11. For example here is the bottom of the *zpool* man page.



It is a bit strange to see Solaris man pages on a GNU/Linux system. However, it turns out that these man pages also have a CDDL license.

Turning now to creating a ZFS storage pool. Similar to Logical Volume Manager (LVM), disk space is not allocated to a specific filesystem; instead it is allocated to a storage pool. Here is where I immediately ran into a problem as my disks use GPT instead of MBR.

```
# gdisk -p /dev/sdb
GPT fdisk (gdisk) version 0.6.13

Usage: gdisk [-l] device_file
[root@ultra fpm]# gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.6.13

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 976773168 sectors, 465.8 GiB
Logical sector size: 512 bytes
```



```
Disk identifier (GUID): 81256E88-5710-42AC-94DE-0E87E38355A4
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 976773134
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048       976773134   465.8 GiB   EF00  EFI System

# zpool create pool1 /dev/sdb
# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_ultra-lv_root
                      51606140  18493680  30491020  38% /
tmpfs                  1027392       124   1027268   1% /dev/shm
/dev/sda2               495844     81949    388295  18% /boot
/dev/sda1               204580      9328    195252   5% /boot/efi
/dev/mapper/vg_ultra-lv_home
                     424395160  10508168 392328928   3% /home
pool1                478937042        21 478937021   1% /pool1

# gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.6.13

Warning! One or more CRCs don't match. You should repair the disk!

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: damaged

****************************************************************************
Caution: Found protective or hybrid MBR and corrupt GPT. Using GPT, but disk
verification and recovery are STRONGLY recommended.
****************************************************************************
Disk /dev/sdb: 976773168 sectors, 465.8 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 05C8E1A6-3D86-F04E-A54E-D2B6B6E12914
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 976773134
Partitions will be aligned on 1-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048       976756750   465.8 GiB   BF01  zfs
   9       976756751       976773134   8.0 MiB     BF07
#
```

As you can see *zpool* changed the size of the partition and corrupted the GPT. However it is still usable so the corruption is minor. I also noticed that *zpool* did not warn me if I was going to overwrite an existing file system.

Next I created a mirrored 100 Gb zpool, created a ZFS file system called *test* and set a 5Gb quota on it.

```
# gdisk /dev/sdb
GPT fdisk (gdisk) version 0.6.13
...

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
...
```



```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048       209717247   100.0 GiB   0700  Linux/Windows data
   2       209717248       419432447   100.0 GiB   0700  Linux/Windows data

Command (? for help): q
# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_ultra-lv_root
                      51606140  18493680  30491020  38% /
tmpfs                  1027392       272   1027120   1% /dev/shm
/dev/sda2               495844     81949    388295  18% /boot
/dev/sda1               204580      9328    195252   5% /boot/efi
/dev/mapper/vg_ultra-lv_home
                     424395160  10509340 392327756   3% /home

# zpool create pool mirror /dev/sdb1 /dev/sdb2

# zfs create pool/test

# zfs list
NAME        USED  AVAIL  REFER  MOUNTPOINT
pool        130K  97.9G    29K  /pool
pool/test    21K  97.9G    21K  /test

# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_ultra-lv_root
                      51606140  18492792  30491908  38% /
tmpfs                  1027392       124   1027268   1% /dev/shm
/dev/sda2               495844     81949    388295  18% /boot
/dev/sda1               204580      9328    195252   5% /boot/efi
/dev/mapper/vg_ultra-lv_home
                     424395160  10507916 392329180   3% /home
pool                 102703004        29 102702975   1% /pool
pool/test            102702996        21 102702975   1% /test

# zfs set quota=5g pool/test

# zfs list
NAME        USED  AVAIL  REFER  MOUNTPOINT
pool        131K  97.9G    29K  /pool
pool/test    21K  5.00G    21K  /test

# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_ultra-lv_root
                      51606140  18492800  30491900  38% /
tmpfs                  1027392       284   1027108   1% /dev/shm
/dev/sda2               495844     81949    388295  18% /boot
/dev/sda1               204580      9328    195252   5% /boot/efi
/dev/mapper/vg_ultra-lv_home
                     424395160  10508820 392328276   3% /home
pool                 102703002        29 102702973   1% /pool
pool/test              5242880        21   5242859   1% /test

# zpool iostat
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
pool         131K  99.5G      0      0     98    230

# zpool history pool
History for 'pool':
2011-01-24.00:14:22 zpool create pool mirror /dev/sdb1 /dev/sdb2
2011-01-24.00:14:53 zfs create pool/test
2011-01-24.00:16:26 zfs set mountpoint=/test pool/test
2011-01-24.00:34:24 zfs set quota=5g pool/test
```



```
# zfs get all pool
NAME  PROPERTY              VALUE                  SOURCE
pool  type                  filesystem             -
pool  creation              Mon Jan 24  0:14 2011  -
pool  used                  131K                   -
pool  available             97.9G                  -
pool  referenced            29K                    -
pool  compressratio         1.00x                  -
pool  mounted               yes                    -
pool  quota                 none                   default
pool  reservation           none                   default
pool  recordsize            128K                   default
pool  mountpoint            /pool                  default
pool  sharenfs              off                    default
pool  checksum              on                     default
pool  compression           off                    default
pool  atime                 on                     default
pool  devices               on                     default
pool  exec                  on                     default
pool  setuid                on                     default
pool  readonly              off                    default
pool  zoned                 off                    default
pool  snapdir               hidden                 default
pool  aclinherit            restricted             default
pool  canmount              on                     default
pool  xattr                 on                     default
pool  copies                1                      default
pool  version               5                      -
pool  utf8only              off                    -
pool  normalization         none                   -
pool  casesensitivity       Not Supported          -
pool  vscan                 off                    default
pool  nbmand                off                    default
pool  sharesmb              off                    default
pool  refquota              none                   default
pool  refreservation        none                   default
pool  primarycache          all                    default
pool  secondarycache        all                    default
pool  usedbysnapshots       0                      -
pool  usedbydataset         29K                    -
pool  usedbychildren        102K                   -
pool  usedbyrefreservation  0                      -
pool  logbias               latency                default
pool  dedup                 off                    default
pool  mlslabel              none                   default
pool  sync                  standard               default
```

All worked as expected.

I have not yet fully tested ZFS on GNU/Linux but so far I am more than happy with it. Some minor glitches and gotchas but such things are to be expected. I am delighted that native ZFS on GNU/Linux is now available and will switch to using it more and more as I gain confidence in it.

Well done, KQ Infotech!

