Interesting Things You Didn't Know You Could Do With ZFS

Allan Jude -- ScaleEngine Inc.

allanjude@freebsd.org twitter: @allanjude

Introduction

Allan Jude

- 13 Years as FreeBSD Server Admin
- FreeBSD src/doc committer (focus: ZFS, bhyve, ucl, xo)
- Co-Author of “*FreeBSD Mastery: ZFS*” and upcoming “*FreeBSD Mastery: Advanced ZFS*” with M. W. Lucas
- Architect of the ScaleEngine CDN (HTTP and Video)
- Host of BSDNow.tv & TechSNAP.tv Podcasts
- Use ZFS for large collections of videos, extremely large website caches, mirrors of PC-BSD pkgs and RaspBSD
- Single Handedly Manage Over 1000TB of ZFS Storage

The Power of ZFS

- Integrated Redundancy (Mirroring, RAID-Z)
- Data Integrity Checking (Checksums, Scrub)
- Pooled Storage (Hot Add Disks)
- Multi-Level Cache (ARC, L2ARC, SLOG)
- Copy-on-Write (no fsck)
- Snapshots and Clones
- Quotas and Reservations
- Transparent Compression (LZ4, GZIP1-9)
- Incremental Replication (zfs send/recv)
- Datasets with Individual Inherited Properties
- Custom Properties
- Fine Grained Delegation

Applying That Power

ZFS has many features, but how can I use them to solve my problems?

ZFS has a very well designed command line user interface, making it very easy for a sysadmin to perform common tasks (add more storage, create new datasets, change settings and properties), accomplish things that were not possible before, as well as extract a great deal more information from the storage system.

ZFS Was Meant To Be Scripted

```
# zfs list -Hp -r -o name,refer,logicalreferenced
sestore5/mysql02    22001288628    24331078144
sestore5/omicron    4822821993     12114904064
sestore5/sevu2/mysql   3562667064     4930608640
# zfs get -Hpr -t filesystem -o name,value compressratio
sestore5/mysql02    2.06x
sestore5/omicron    9.37x
sestore5/sevu2/mysql   2.97x
# zfs list -Hd 1 -o name -s used
zroot/usr
zroot/var
zroot/bootenv
```

An Actual Script

```
# zfs list -Hprt filesystem -o name,refer,logicalreferenced
while read dataset used logical; do
    diff=$(($logical - $used))
    percent=$(($diff * 100 / $logical * 100 / 100))
    if [ $diff -gt 0 ]; then
        printf "${dataset}\t${diff}\t${percent}%%\n";
    fi
done
```

```
Dataset               Bytes Saved    Percent
sestore5/mysql02      2329603330     9%
sestore5/omicron      7291274343     60%
sestore5/sevu2/mysql  1367950734     27%
```

Abusing Properties and Inheritance

```
# zfs create -o exec=off \
    -o mountpoint=/usr/local zroot/local
# zfs create -o canmount=off \
    -o mountpoint=/usr/local zroot/local_bin
# zfs create zroot/local_bin/bin
# zfs create zroot/local_bin/sbin
# zfs create -o mountpoint=/usr/local/etc/rc.d \
    zroot/local_bin/rc.d
```

Now Install your files, and lock it down

```
# zfs set readonly=on zroot/local_bin
```

The Result:

```
# zfs list -o name,mountpoint,readonly
NAME                  MOUNTPOINT            RDONLY
zroot/local           /usr/local            off
zroot/local_bin       /usr/local            on **
zroot/local_bin/bin   /usr/local/bin        on
zroot/local_bin/rc.d  /usr/local/etc/rc.d   on
zroot/local_bin/sbin  /usr/local/sbin       on
```

Files in the bin, rc.d, and sbin directories cannot be modified, while files in the other directories cannot be executed. One hierarchy, two parents to inherit from.

** This dataset has canmount=off, so is never actually mounted. It serves only as a parent to the child datasets.

Delegation

The best way to get more work done, is to delegate the work to someone else.

DOGBERT THE CEO
WE'LL EXECUTE OUR STRATEGY IN THE USUAL WAY.
www.dilbert.com scottadams@aol.com
THE POWERFUL WILL DELEGATE TO THE UNTALENTED UNTIL FAILURE IS ACHIEVED.
6-11-09 © 2009 Scott Adams, Inc./Dist. by UFS, Inc.
HOW LONG WILL THAT TAKE?
WE JUST FINISHED.

ZFS Allows you to delegate fine-grained permissions to unprivileged users, so they can control their own data, and stop pestering you.

Unwashed Users...

Most sysadmin try to avoid dealing with “Users”. For this example, let us say the user’s name is Lucas. Lucas keeps pestering you to create snapshots of his home directory for him. As a lazy sysadmin, this annoys you. Give Lucas the ability to create his own snapshots, but only of his home directory dataset, and nothing else:

```
# zfs allow -u lucas snapshot,rollback zroot/usr/home/lucas
# zfs allow zroot/usr/home/lucas
---- Permissions on zroot/usr/home/lucas -----
Local+Descendent permissions:
	user lucas rollback,snapshot
```

Self Service == Happy Sysadmin

```
$ zfs snapshot zroot/usr/home/lucas@chapter2
$ zfs list -t all -r -o name,used zroot/usr/home/lucas
NAME                              USED
zroot/usr/home/lucas              96K
zroot/usr/home/lucas@chapter2     0
$ zfs snapshot zroot@naughty
cannot create snapshots : permission denied
```

Destroying snapshots and creating clones requires the ‘mount’ permission. In order for users to be able to mount: vfs.usermount=1

Delegation Inheritance

The delegation system allows you to specify if a delegation is ‘Local’ (Applies only to this dataset), ‘Decendant’ (Applies only to child datasets), or both (the default).

I want to allow Lucas to destroy the snapshots he created, but not shoot himself in the foot and destroy his entire home directory.

Or maybe I want to allow him some permissions only on the datasets he creates (‘sticky bit’ for delegation), rather than the entire tree.

Avoid Foot Shooting

```
# zfs allow -d lucas destroy,mount zroot/usr/home/lucas
# zfs allow zroot/usr/home/lucas
---- Permissions on zroot/usr/home/lucas -------------------
Descendent permissions:
     user lucas destroy,mount
Local+Descendent permissions:
     user lucas rollback,snapshot
$ zfs destroy -v zroot/usr/home/lucas/desc
will destroy zroot/usr/home/lucas/desc
```

Sticky Delegation

```
# zfs allow lucas create,mount zroot/usr/home/lucas
# zfs allow -c destroy zroot/usr/home/lucas
# zfs allow zroot/usr/home/lucas
---- Permissions on zroot/usr/home/lucas ----------------------
Permission sets:
     destroy
Local+Descendent permissions:
     user lucas create,mount,rollback,snapshot
$ zfs create zroot/usr/home/lucas/new
$ zfs destroy -v zroot/usr/home/lucas/new
will destroy zroot/usr/home/lucas/new
```

Lucas == Evil

However, what if Lucas is incompetent, or evil?

He creates a crontab that takes a snapshot every minute, or tries to create millions of snapshots, bogging the system down?

```
# zfs set snapshot_limit=3 zroot/usr/home/lucas
$ zfs snapshot zroot/usr/home/lucas@three
$ zfs snapshot zroot/usr/home/lucas@four
cannot create snapshot 'zroot/usr/home/lucas':
out of space
```

Jails

Of course, I distrust Lucas too much to allow him to have a regular user account on my server. Instead, I lock him up in a jail, where I can let him run rabid without worry.

```
# zfs create -o jailed=on -o mountpoint=/jail \
    zroot/jails/lucas/jail
# jail -c path=/zroot/jails/lucas \
    mount.devfs allow.mount allow.mount.zfs \
    host.hostname=lucas ip4.addr=127.0.0.2 \
    exec.poststart = \
    "/sbin/zfs jail lucas zroot/jails/lucas/jail"; \
    command=/bin/sh
```

Jailing a Dataset

```
# jexec lucas sh
jail# zfs create zroot/jails/lucas/jail/foo
jail# zfs list -o name,mountpoint
NAME                         MOUNTPOINT
zroot                        /zroot
zroot/jails                  /zroot/jails
zroot/jails/lucas            /zroot/jails/lucas
zroot/jails/lucas/jail       /jail
zroot/jails/lucas/jail/foo   /jail/foo
```

Containers Are Good

Now Lucas can do whatever he needs to do, set mount points, create and destroy datasets, delegate datasets to regular users inside the jail, etc.

The only thing he cannot do, is change the quota on a dataset, as this might allow him to gain access to more space than I have allocated to him. Otherwise, from inside the jail, it is hard to tell that you don't control the entire zpool.

Now if only it was that easy to keep ~~Lucas~~ users from breaking things in the real world.

How do you juggle many patches?

```
# zfs create -p mypool/svn/base
# svn checkout …
# zfs snapshot mypool/svn/base@r251234
# zfs clone mypool/svn/base@r251234 mypool/svn/ifconfig_err
# zfs clone mypool/svn/base@r251234 mypool/svn/vmstat_libxo
# svn commit …
# zfs destroy mypool/svn/ifconfig_err
```

Traveling with WIP

… Write code at home

```
# zfs snapshot home/svn@before_vbsdcon
# ssh allan@home zfs send -Rv
home/svn@before_vbsdcon | zfs receive mobile/svn
```

… Write code at the conference

```
# zfs snapshot mobile/svn@after_vbsdcon
# zfs send -Rv -I before_vbsdcon
mobile/svn@after_vbsdcon | ssh allan@home zfs
receive home/svn
```

… Finish code at home

```
# svn commit ...
```

ZFS Diff

ZFS lets you determine the differences between a live dataset and a snapshot, or between two snapshots. Much quicker than rsync et all to determine just which files have been modified, rather than stat()’ing every file.

Also useful for forensics. What changed in this dataset after the web server jail was compromised? That sshd binary definitely shouldn’t have changed...

Finding Out What Happened

```
# zfs diff zroot/bootenv/default@before_freebsd_update_p0
M       /boot/kernel/ahd.ko  (+1)
        …
M       /boot/kernel/geom_concat.ko  (+1)
+   /lib/libbsdxml.so.4
+   /bin/freebsd-version
+   /rescue/sync
+   /usr/bin/netstat
+   /usr/lib/libbsdxml.a
+   /usr/lib/libbsdxml_p.a
+   /usr/sbin/vidcontrol
```

Poking The Innards

A number of utilities and commands exist to get at the real innards of ZFS. Quick overview:

- zfs send … | zstreamdump
- zdb -mmm … (metaslab free, fragmentation)
- zdb -dddd ds obj (details of obj# in dataset)
- zdb -ii pool (what is in your ZIL)
- zdb -bbb (search for free space leaks and breakdown space usage by object type)
- zpool clear (reset the checksum and read/write error counters)

Store Configuration in Properties

ZFS User Properties are an interesting place to store configuration. If you jail, VM, or iSCSI target stores its configuration as properties of the ZFS dataset (or zvol), the settings move with the dataset when it is replicated.

Currently, it seems that dataset properties are limited to 8kb of storage each, although it is not clear if this limit is intentional

bhyve config on a zvol

```
# zfs get -o property,value all zroot/vm0001
bhyve:cpus          4
bhyve:memory        2048
bhyve:console       /dev/nmdm0001A
bhyve:networks      [{type:virtio-net,name:tap7}]
bhyve:disks   [ { type:ahci-hd,
    path:/dev/zvol/zroot/disk0001 }, {
    type:ahci-hd, path:/dev/zvol/zroot/disk0002 }]
```

Lightning Round

- zfs create -p (like mkdir -p)
- zfs rename -u (do not remount, yet)
- zfs destroy -rvn dataset@fromsnap%tosnap (estimate how much space will be freed by recursively removing this range of snapshots)
- zfs rename -r fromsnap tosnap (rename a set of recursive snapshots)
- zfs inherit -S (unset received properties)
- zfs get usedby{dataset,snapshots,children,refresv}
- zfs set volmode=dev p/ds (doesn't apply until the device is recreated @ boot)
  - zfs rename p/vol p/vol1; zfs rename p/vol1 p/vol
- zfs create -o casesensitivity=insensitive (Mac/Win)

Upcoming Features in ZFS

A lot of new things are coming to ZFS "Soon"™

- Resumable Send
- Replication Record Checksums
- Receive Prefetch
- Compressed ARC
- Persistent L2ARC
- VDEV Removal (Stripe and Mirrors Only)
- Allocation Throttle (Write Bandwidth)
- New Prefetch Algorithms
- More Checksums (SHA512, Skein, Edon-R)

More Upcoming ZFS Features

- Compressed Send
  - Current Process: Uncompress, Transmit, Compress
  - New Process: Transmit, Uncompress, Compress
  - Since network is usually bottleneck, this achieves better performance. Maybe it could avoid the recompress step as well, if algos are the same
- Channel Programs (Multiple Commands In a Single Atomic Transaction)
- Fast File Cloning
- Dedup Throttle (and Dedicated DDT L2ARC)
- JSON CLI Output

Future Ideas for ZFS

- Replication Rebase (Catch up to a newer snapshot without a common snapshot)
- HA/FO Features (Multi-Modifier Protection)
- Clustered Features
- Tiered Storage (NVMe/SSD -> HDD -> Tape)
- Compatibility with Host-Aware SMR (Shingled Magnetic Recording) Drives
- Per dataset I/O stats
- Per dataset or jail/zone I/O QoS
- Separate VDEVs for Metadata, or Datasets

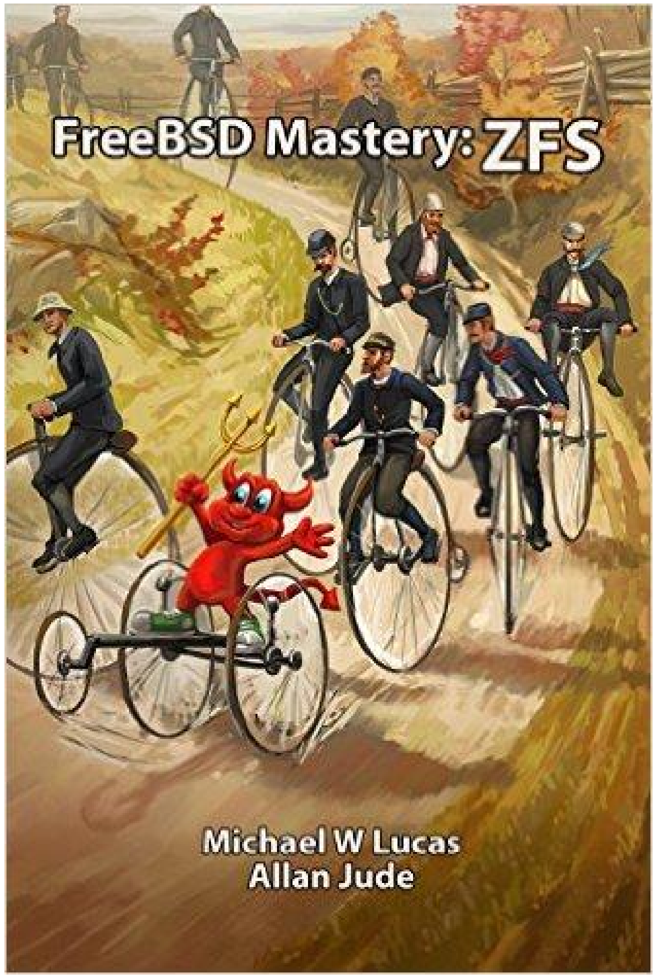
OpenZFS Developers Summit

The third annual OpenZFS Developer Summit will be held in San Francisco, October 19 - 20, 2015. This will be the 10th anniversary of the open sourcing of ZFS! All OpenZFS developers are invited to participate.

First day: Presentations and Talks, followed by Beer Bash and Dinner

Second Day: Hackathon with prizes

The Book



- 188 pages of easy reading
- DRM-Free ebook or Print Edition
- How hardware affects ZFS
- Configure datasets
- Repair and monitor storage pools
- Expand your storage
- How compression enhances performance
- How copy-on-write works
- Understand how ZFS uses and manages space
- Buy it at ZFSBook.com

The Next Book

FreeBSD Mastery: Advanced ZFS

Still being written, will cover more depth and internals. How to deal with pathological performance, advanced features, and what to do when things go wrong.

If you have ever had trouble with ZFS, tell us about it, so we can add it to the book.

Look for it before the end of the year.

Podcasts

BSDNow.tv is a weekly video podcast featuring News, Interviews and Tutorials about the BSD family of Operating Systems. Hosted by Kris Moore (founder of PC-BSD) and Myself.

TechSNAP.tv is a weekly sysadmin video podcast covering an OS agnostic range of security and production issues for those working, studying or interested in the sysadmin / devops / infosec field.

Twitter: @allanjude Email: allanjude@freebsd.org