Bulk data storage with FreeBSD and ZFS in a mixed environment

Ian Clark

Computing & Imaging Systems Manager

Department of Genetics, School of Biological Sciences

25th June 2014
Or
How I learned to stop worrying and love the JBOD

Introduction
ZFS and zpools
  Comparisons
  Features
  Practical Example
Caveats
FreeBSD in brief

SFTP
NFS
  Server
  Clients
CIFS
  Stand-alone Samba server
  AD Domain Member
Introduction

- There’s always more data
- Object storage is great, but it’s a painful transition from standard filesystems.
- To provide a standard filesystem, we need a big block of disk.
  - Hardware RAID is fast but fragile and inflexible.
  - Software RAID is more flexible and about as fast.
- Or what if the filesystem was aware of multiple disks?

![Figure 1: Size of quarterly full backups](image)
Comparisons

- Object storage expects clients (or a gateway) to talk to multiple servers & disks themselves.
Comparisons

- Object storage expects clients (or a gateway) to talk to multiple servers & disks themselves.
- RAID attempts to give something that looks like a big hard disk.
Comparisons

- Object storage expects clients (or a gateway) to talk to multiple servers & disks themselves.
- RAID attempts to give something that looks like a big hard disk.
- ZFS provides a filesystem (more or less) directly.
Features

- Checksums. Checksums everywhere.
Features

▶ Checksums. Checksums everywhere.
▶ On-disk data always consistent: Copy on Write.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
- Slab-based allocation.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
- Slab-based allocation.
- Intelligent caching.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
- Slab-based allocation.
- Intelligent caching.
- Quotas.
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
- Slab-based allocation.
- Intelligent caching.
- Quotas.
- NFSv4 ACLs (more or less a copy of NTFS ACLs.)
Features

- Checksums. Checksums everywhere.
- On-disk data always consistent: Copy on Write.
- Snapshots and clones.
- Multi-disk redundancy.
- Dynamic striping
- Slab-based allocation.
- Intelligent caching.
- Quotas.
- NFSv4 ACLs (more or less a copy of NTFS ACLs.)
- Filesystem streaming, including incremental streams.
Practical Example

Bought in two phases:

**Nov 2010** Server with 8 2TB SATA disks, 3ware controller, 24GB RAM

**Jul 2012** JBOD, better controller, nearline SAS disks, more RAM, SSDs

<table>
<thead>
<tr>
<th>Server Chassis</th>
<th>Supermicro SC836 3U 16 disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motherboard</td>
<td>Supermicro X8DT6</td>
</tr>
<tr>
<td>Processor</td>
<td>Dual Xeon E5620</td>
</tr>
<tr>
<td>RAM</td>
<td>64 gigabytes DDR3 1066MHz</td>
</tr>
<tr>
<td>Host Bus Adaptors</td>
<td>LSI SAS 9200-8i (Internal backplane)</td>
</tr>
<tr>
<td></td>
<td>LSI SAS 9200-8e (JBOD backplanes)</td>
</tr>
<tr>
<td>JBOD chassis</td>
<td>Supermicro SC847 4U 45 disk JBOD</td>
</tr>
<tr>
<td>Cache</td>
<td>2x Intel SSD 320 80GB</td>
</tr>
<tr>
<td>Disks</td>
<td>60x Seagate Constellation 2TB ES.2</td>
</tr>
</tbody>
</table>
Caveats

▶ ZFS

- Deduplication can bring your server to its knees.
- Performance drops when a set of disks is nearly filled.
- Data isn’t redistributed when new sets of disks are added.
- FreeBSD
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.

- It’s not actually magic
- You still need off-site backups.
- Good idea to run smartd too.
Caveats

- **ZFS**
  - Deduplication can bring your server to its knees.
Caveats

- ZFS
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
Caveats

- ZFS
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
  - Data isn’t redistributed when new sets of disks are added.

- FreeBSD
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.

- It’s not actually magic
  - You still need off-site backups.
  - Good idea to run smartd too.
Caveats

- **ZFS**
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
  - Data isn’t redistributed when new sets of disks are added.
- **FreeBSD**
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.
  - It’s not actually magic
  - You still need off-site backups.
  - Good idea to run smartd too.
Caveats

▶ **ZFS**
  ▶ Deduplication can bring your server to its knees.
  ▶ Performance drops when a set of disks is nearly filled.
  ▶ Data isn’t redistributed when new sets of disks are added.

▶ **FreeBSD**
  ▶ It’s difficult to map slots on JBODs to drives.
Caveats

▶ ZFS
  ▶ Deduplication can bring your server to its knees.
  ▶ Performance drops when a set of disks is nearly filled.
  ▶ Data isn’t redistributed when new sets of disks are added.

▶ FreeBSD
  ▶ It’s difficult to map slots on JBODs to drives.
  ▶ Disk failure detection could be better.
Caveats

▶ ZFS
  ▶ Deduplication can bring your server to its knees.
  ▶ Performance drops when a set of disks is nearly filled.
  ▶ Data isn’t redistributed when new sets of disks are added.

▶ FreeBSD
  ▶ It’s difficult to map slots on JBODs to drives.
  ▶ Disk failure detection could be better.
  ▶ Some controllers lose disks behind SAS expanders at random.
Caveats

- **ZFS**
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
  - Data isn’t redistributed when new sets of disks are added.

- **FreeBSD**
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.

- It’s not actually magic
Caveats

- **ZFS**
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
  - Data isn’t redistributed when new sets of disks are added.

- **FreeBSD**
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.

- **It’s not actually magic**
  - You still need off-site backups.
Caveats

- **ZFS**
  - Deduplication can bring your server to its knees.
  - Performance drops when a set of disks is nearly filled.
  - Data isn’t redistributed when new sets of disks are added.

- **FreeBSD**
  - It’s difficult to map slots on JBODs to drives.
  - Disk failure detection could be better.
  - Some controllers lose disks behind SAS expanders at random.

- **It’s not actually magic**
  - You still need off-site backups.
  - Good idea to run smartd too.
FreeBSD in brief

- It’s a UNIX
- Very lightweight base system
- Third party software can be installed by three routes:
  - Packages: Pre compiled `pkg install bash`
  - Ports: source `cd /usr/ports/net/samba4 && make install`
  - Traditional: Download and compile the source yourself
- Third party software ends up in `/usr/local/`
- Most non-service specific config in `/etc/rc.conf`

Curious?

If you want more history, specifics see [http://www.freebsd.org](http://www.freebsd.org)
SFTP

- Very simple, probably running already
- Quite secure, becoming very secure with good key management
- Rarely blocked on the public Internet
- Doesn’t (reliably) give access to ACLs

Security

You probably don’t want non-administrative users to log in and run commands. At the bottom of `/etc/ssh/sshd_config` append

```
Match User *,!co
  ForceCommand /usr/libexec/sftp-server
```

If you use passwords it’s worth running ”Denyhosts” or similar to block brute force guessing.
NFS Server

- Version 3 spoken by almost all UNIX systems
- Trivial to set up
- You’ve got to trust your clients explicitly
- Version 4 a bit better, uses Kerberos

NFSv3 quick start

Add `nfs_server_enable="YES"` to `/etc/rc.conf`

```
zfs set sharenfs="-maproot=nobody client.hostname" pool/home
zfs set sharenfs="-mapall=nobody -ro -network 10/8" pool/public
```

More sharenfs options documented in the `exports(5)` manpage.
NFS Clients

- Standard model for ZFS is one file system per user/group
- You’ll probably want to use autofs
- I use autofs 5 on Debian

/etc/auto.home

* server.hostname:/pool/home/&

/etc/auto.master

/home /etc/auto.home

mount output (trimmed)

/etc/auto.home on /home type autofs
server:/pool/home/user on /home/user type nfs
[global]
workgroup = EXAMPLE
security = user
# Samba’s wrapper for FreeBSD’s kqueue has a bug
kernel change notify = no
[share]
comment = A share
browseable = yes
writable = yes
vfs objects = zfsacl
nfs4:mode = special
nfs4:chown = yes
zfsacl:acesort = dontcare
AD Domain Member: Overview

1. Ensure ports & packages are up to date
2. Install package “net/samba4”
3. Remove it (but not its dependencies)
4. Rebuild the port, ensuring “experimental modules” are enabled
5. Create a new /usr/local/etc/smb4.conf
6. Create a computer account in the domain
7. Enable & Start the services
8. Configure Name Services & Pluggable Authentication Modules
AD Domain Member: Configuration

```
[global]
    workgroup = AD
    realm = AD.EXAMPLE.COM
    security = ads
    winbind enum groups = yes
    winbind enum users = yes
    winbind nss info = rfc2307
    idmap config * : backend = tdb
    idmap config * : range = 1000000-19999999
    idmap config AD : schema_mode = rfc2307
    idmap config AD : backend = ad
    idmap config AD : range = 1000-50000
    kernel change notify = no
```
[homes]
  comment = Home Directories
  browseable = yes
  writable = yes
  hide files = ./*/desktop.ini/$RECYCLE.BIN/
  vfs objects = zfsacl
  nfs4:mode = special
  nfs4:chown = yes
  zfsacl:acesort = dontcare
  root preexec = /usr/local/bin/updatehome.pl '%%U'
1. Join the domain:
   # net ads join -U Administrator
   Administrator@AD.EXAMPLE.COM’s password:
   Using short domain name -- AD
   Joined 'SERVER' to realm 'AD.EXAMPLE.COM'
   DNS update failed!

2. Enable services:
   # echo 'samba_server_enable="YES"' >> /etc/rc.conf
   # echo 'winbindd_enable="YES"' >> /etc/rc.conf

3. Start services:
   # service samba_server start

4. Test it:
   # wbinfo -P
   checking the NETLOGON dc connection to "dc0.ad.example.com" succeeded
Winbindd is a service that acts as a shim between the UNIX standard authentication/user database functions and Active Directory.

- **User database:** `/etc/nsswitch.conf`. Comment out:
  ```
  passwd: compat
  and
  group: compat
  ```
  Add:
  ```
  passwd: files winbind
  group: files winbind
  ```

- **Authentication:** `/etc/pam.d/ssh`. Add:
  ```
  auth sufficient /usr/local/lib/pam_winbind.so
  ```
  in the auth section.
Future work & Thanks

Future work

▶ Test FreeNAS
▶ Use ZFS snapshot streaming for backups
▶ Samba 4 directory services
▶ Test ZFS on Linux

Thanks

▶ Transec & ANS for supplying what I ask for
▶ Paul Sumption for off-site rack space
▶ Phase one paid from departmental funds
▶ Phase two paid for by the Isaac Newton Trust
▶ LaTeX editing http://www.writelatex.com