



Software Freedom Day

Overview, ZFS & Zones

Alan Hargreaves
Principal Field Technologist
Sun Microsystems

What I'll discuss

- Who am I?
- What I won't discuss
- Solaris
 - Some background
- OpenSolaris
 - What is it? Why do it? ...
- ZFS
- Zones
- A brief demo

Who am I?

Name: Alan Hargreaves

Title: Principal Field Technologist

Time at Sun: 10 years

My Work: Performance,
Kernel,
Interoperability,
magnet for ugly calls

My Time: 10 y.o. Boy/Girl twins,
writing/performing music



What I won't discuss



Solaris

- Solaris 2.0 – 1992
- Collaboration between AT&T and Sun
- Based on SysVr4
- Solaris 10 – January 31, 2005
 - Update 8 due out shortly
- Widely used in the enterprise sphere

OpenSolaris - History

- DTrace – January 25, 2005
- OS/Net – June 14, 2005
 - Buildable/bootable
- Project Indiana – Ian Murdoch, March 2007
- OpenSolaris Distribution based on Indiana
 - 2008.05
 - 2008.11
 - 2009.06

OpenSolaris – Why do it?

- No-one has a monopoly on “good ideas”
- A good codebase
- The big push to Open Source came from Engineering
- Sun's history was in Open code, it's disappointing that we didn't do this far earlier
- Allow anyone to work with and develop the cutting edge of Solaris Development

OpenSolaris – Features today

- DTrace, ZFS, Crossbow, Zones, more friendly user environment, tracking gnome releases, image packaging system, time-slider, xVM, COMSTAR, CIFS, x86/x64/SPARC, NWAM, MySQL, RBAC, Crypto Framework, Trusted Extensions, FMA, Clustering, Support, ...



ZFS Overview

○ Pooled storage

- Completely eliminates the notion of volumes
- Does for storage what VM did for memory

○ Transactional object system

- Always consistent on disk – no fsck, ever
- Universal – file, block, iSCSI, swap ...

○ Provable end-to-end data integrity

- Detects and corrects silent data corruption
- Historically considered “too expensive” – no longer true

○ Simple administration

- Concisely express your intent

ZFS – Trouble with existing filesystems

- ◉ No defense against silent data corruption
 - Any defect in disk, controller, cable, driver, laser, or firmware can corrupt data silently; like running a server without ECC memory
- ◉ Brutal to manage
 - Labels, partitions, volumes, provisioning, grow/shrink, /etc files...
 - Lots of limits: filesystem/volume size, file size, number of files, files per directory, number of snapshots ...
 - Different tools to manage file, block, iSCSI, NFS, CIFS ...
 - Not portable between platforms
- ◉ Dog slow
 - Linear-time create, fat locks, fixed block size, naïve prefetch, dirty region logging, painful RAID rebuilds, growing backup time

ZFS Objective

End the suffering

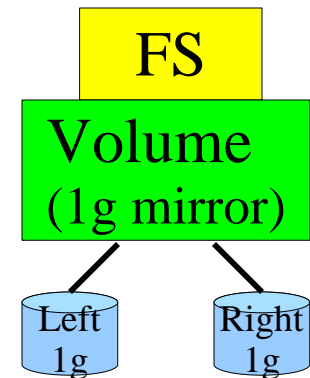
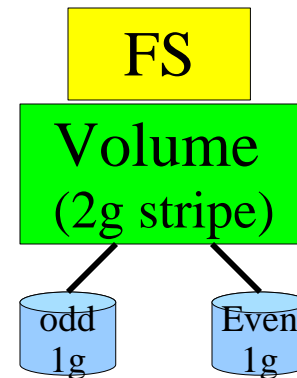
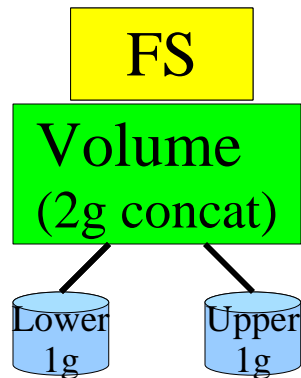
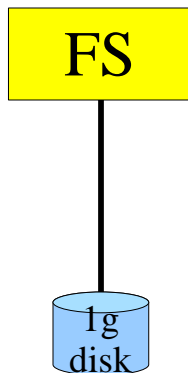
- Figure out why storage has gotten so complicated
- Blow away 20 years of obsolete assumptions
- Design an integrated system from scratch

Why volumes exist

In the beginning each filesystem managed a single disk.

It wasn't very big.

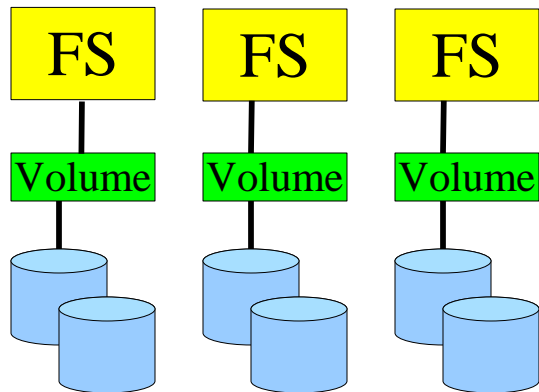
- Customers wanted more space, bandwidth, reliability
 - Hard: redesign filesystems to solve these problems well
 - Easy: insert a shim (“volume”) to cobble disks together
- An industry grew up around the FS/volume model
 - Filesystem, volume manager sold as separate products
 - Inherent problems in FS/volume interface can't be fixed



FS/Volume Model vs. Pooled Storage

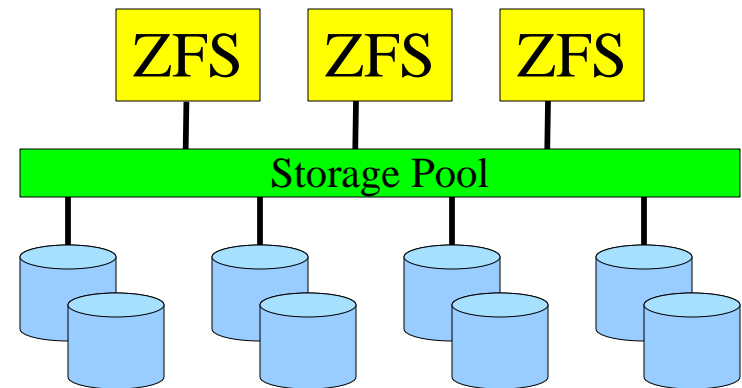
Traditional Volumes

- Abstraction: virtual disk
- Partition/volume for each FS
- Grow/shrink by hand
- Each FS has limited bandwidth
- Storage is fragmented, stranded



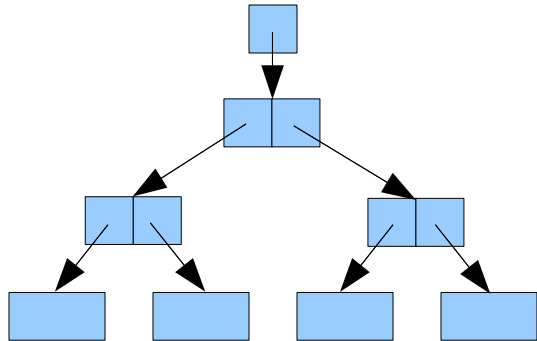
ZFS Pooled Storage

- Abstraction: malloc/free
- No partitions to manage
- Grow/shrink automatically
- All bandwidth always available
- All storage in the pool is shared

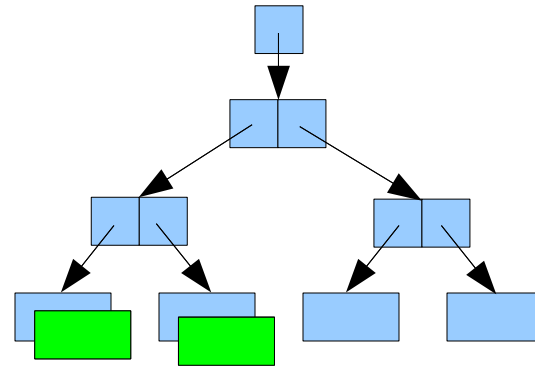


Copy on Write Transactions

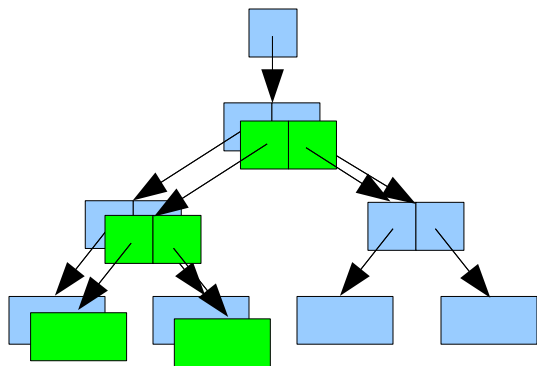
1. Initial Block Tree



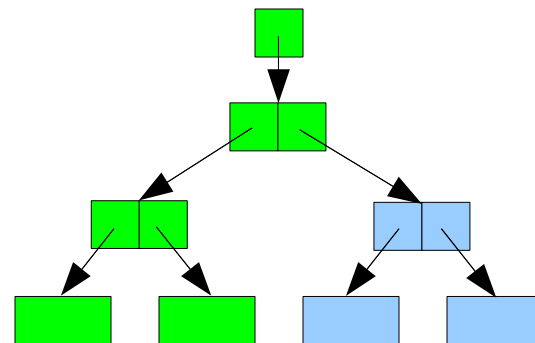
2. COW some blocks



3. COW indirect blocks

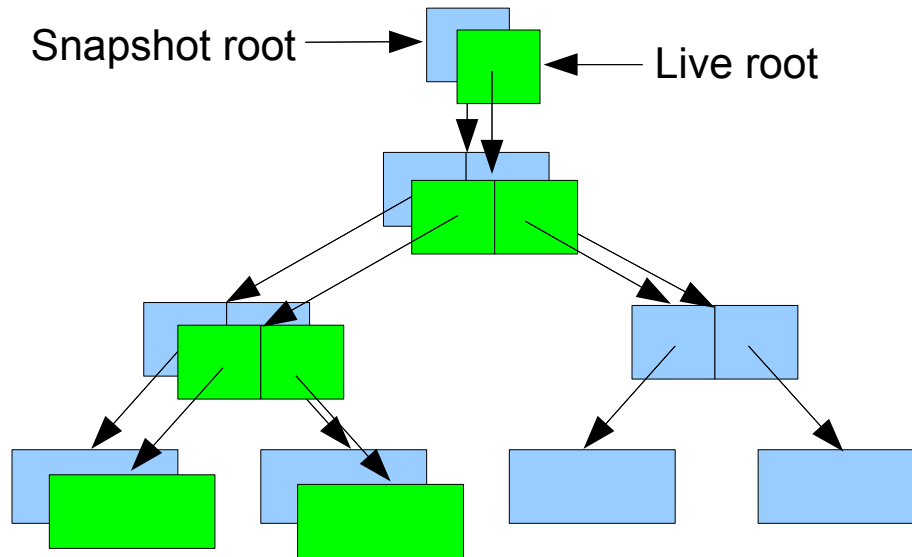


4. Rewrite uberblock (atomic)



Bonus – Constant time snapshots

- At end of TX group, don't free COWed blocks
- Actually cheaper to take a snapshot than not!



- The tricky part: how do you know when a block is free?

End-to-End data integrity in ZFS

Disk Block Checksums

- Checksum stored with data block
- Any self-consistent block will pass
- Can't detect stray writes
- Inherent FS/volume interface limitation

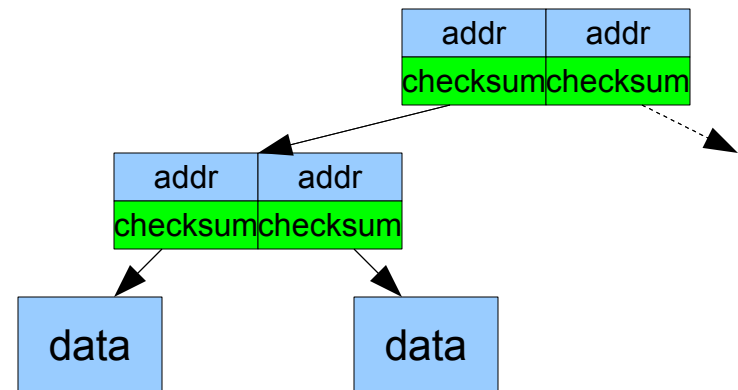


Disk checksum only validates media

✓	Bit rot	✓
✗	Phantom writes	✓
✗	Misdirected reads & writes	✓
✗	DMA Parity	✓
✗	Driver Bugs	✓
✗	Accidental overwrite	✓

ZFS Data Authentication

- Checksum stored in parent block pointer
- Fault isolation between data and checksum
- Checksum hierarchy forms self-validating Merkle tree



ZFS validates the entire I/O Path

Creating pools and filesystems

- Create a mirrored pool named “tank”

```
# zpool create tank mirror c2d0 c3d0
```

- Create home directory filesystem, mounted at /export/home

```
# zfs create tank/home  
# zfs set mountpoint=/export/home tank/home
```

- Create home directories for several users

Note: automatically mounted at /export/home/{ahrens,bonwick,billm} thanks

```
# zfs create tank/home/ahrens  
# zfs create tank/home/bonwick  
# zfs create tank/home/billm
```

- Add more space to the pool

```
# zpool add tank mirror c4d0 c5d0
```

Zones Overview

- Virtualises O/S services for running applications
- Provides Isolation from the rest of the system
 - processes running in one zone cannot see or impact processes in other zones
 - even with superuser credentials in the non-global zone
- Provides an abstraction layer from the physical attributes of the machine
 - eg physical device paths
- Useful for consolidation and improving utilisation
- Requires Solaris 10 or later (e.g. OpenSolaris)
- Max number of zones is 8192 (actual determined by resource requirements)

How Zones Work

○ Non-global zone looks like a box

- Can run one or more applications
- Doesn't interact with the rest of the system
- Applications that are running in the same instance of the Solaris Operating System can then be managed independently of one other.
- Different versions of the same application can be run in different zones
- Processes can only communicate with other processes in the same zone without using network APIs
 - Shared network stacks
 - Exclusive network stacks

How Zones Work (2)

○ Global Zone

- Can see all processes in the machine
- Unprivileged processes may impact non-global zones
- Should only be used for administration when non-global zones are present

○ Each zone has a name and an id

- Global zone is always “global” with ID 0
- The ID is assigned a zone boot time, so they can change!
- Node names are independent of zone names

○ Each zone has it's own

- Path to it's root directory relative to the global root
- It's own scheduling class, and resource controls, if so desired

How zones are administered

- A **global administrator** has the Primary Administrator role. When logged in to the global zone, the global administrator can monitor and control the system as a whole.
- A non-global zone can be administered by a **zone administrator**. The global administrator assigns the Zone Management profile to the zone administrator. The privileges of a zone administrator are confined to a non-global zone.

Creating a zone

A simple example

```
$ pfexec zonecfg -z myzone
myzone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:myzone> create
zonecfg:myzone> set zonepath=/rpool/zones/myzone
zonecfg:myzone> set autoboot=true
zonecfg:myzone> verify
zonecfg:myzone> commit
zonecfg:myzone> exit
$ pfexec zoneadm -z myzone install
A ZFS file system has been created for this zone.
  Publisher: Using opensolaris.org (http://pkg.opensolaris.org/dev/).
  Image: Preparing at /rpool/zones/myhome/root.
  Cache: Using /var/pkg/download.  Sanity Check: Looking for 'entire' incorporation.
  Installing: Core System (output follows)
No updates necessary for this image.      Installing: Additional Packages (output follows)
DOWNLOAD                                PKGS      FILES      XFER (MB)
Completed                                32/32     3192/3192   19.4/19.4
PHASE                                     ACTIONS
Install Phase                             4137/4137
  Note: Man pages can be obtained by installing SUNWman
Postinstall: Copying SMF seed repository ... done.
Postinstall: Applying workarounds.
  Done: Installation completed in 322.582 seconds.

Next Steps: Boot the zone, then log into the zone console
             (zlogin -C) to complete the configuration process
$ pfexec zoneadm -z myzone boot
```

Creating a zone (2)

- On the first boot you need to follow the prompts to install the zone
- Installing clones is the same except for the zoneadm install command.
- Imagine we did exactly the same config for myclone.

```
$ pfexec zoneadm -z myclone clone myzone
```

- This took about 3 seconds

Resources

○ OpenSolaris

- <http://opensolaris.org>

○ ZFS

- http://opensolaris.org/os/community/zfs/docs/zfs_last.pdf
- <http://opensolaris.org/os/community/zfs/>

○ Zones

- <http://dlc.sun.com/osol/docs/content/SYSADRM/zones.intro-1.html>
- http://blogs.sun.com/observatory/entry/cloning_zones

○ Me

- <http://blogs.sun.com/tpenta>