ZFS — and why you need it

Nelson H. F. Beebe and Pieter J. Bowman

University of Utah
Department of Mathematics
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA

Email: beebe@math.utah.edu, bowman@math.utah.edu

15 February 2017

What is ZFS?

- Zettabyte File System (ZFS) developed by Sun Microsystems from 2001 to 2005, with open-source release in 2005 (whence OpenZFS project): SI prefix zetta $\equiv 1000^7 = 10^{21}$
- Sun Microsystems acquired in 2010 by Oracle [continuing ZFS]
- ground-up brand-new filesystem design
- exceptionally clean and well-documented source code
- enormous capacity
  - $2^8 \approx 255$ bytes per filename
  - $2^{48} \approx 10^{14}$ files per directory
  - $2^{64} \approx 10^{18}$ bytes per file [1 exabyte]
  - $2^{78} \approx 10^{23} \approx \frac{1}{2}$ Avogadro's number bytes per volume
- disks form a **pool** of storage that is always consistent on disk
- disk blocks in pool allocatable to any filesystem using pool
- [relatively] simple management
- optional dynamic quota adjustment
- ACLs, snapshots, clones, compression, encryption, deduplication, case-[in]sensitive filenames, Unicode filenames, ...

Why use ZFS?

- ZFS provides a stable flexible filesystem of essentially unlimited capacity [in current technology] for decades to come
- we have run ZFS under Solaris for 11+ years, with neither data loss, nor filesystem corruption
- easy to implement *n*-way live mirroring [*n* up to 12 (??limit??)]
- snapshots, even in large filesystems, take only a second or so
- optional hot spares in each storage pool
- with ZFS *zpool import*, a filesystem can be moved to a different server, even one running a different O/S, as long as ZFS feature levels permit
- ZFS filesystems can be exported via FC, iSCSI, NFS (v2–v4) or SMB/CIFS to other systems, **including** those without native support for ZFS
- blocksize can be set in powers-of-two from $2^9 = 512$ to $2^{17} = 128$K or with *large_blocks* feature, to $2^{20} = 1$M; default on all systems is 128K.
- small files are stored in 512-byte sub-blocks of disk blocks

Where do we run ZFS?

- Solaris ⇐ *main filesystem for 10,000+ users*
- Dyson ⇐ *fork of illumos and OpenSolaris with Debian GNU toolset*
- FreeBSD
- FreeNAS and TrueNAS ⇐ *products of iXsystems*
- GhostBSD ⇐ *fork of FreeBSD 10.3*
- GNU/Linux CentOS ⇐ *unsupported Red Hat*
- GNU/Linux Debian
- GNU/Linux Ubuntu
- Hipster ⇐ *rolling update of OpenIndiana*
- Illumian ⇐ *fork of OpenSolaris 11 illumos*
- Mac OS X ⇐ *from OpenZFS, not from Apple*
- OmniOS ⇐ *fork of OpenSolaris 11 illumos*
- OpenIndiana ⇐ *fork of OpenSolaris 11 illumos*
- PC-BSD ⇐ *fork of FreeBSD 10.3*
- Tribblix ⇐ *fork of illumos, OpenIndiana, and OpenSolaris*
- TrueOS ⇐ *rolling-update successor to PC-BSD*
- XStreamOS ⇐ *fork of OpenSolaris 11 illumos*

zfs subcommands

allow
clone
create
destroy
get
groupspace
inherit
mount
promote
receive
rename
rollback
send
set
share
snapshot
unallow
unmount
upgrade
userspace

```
# zfs snapshot tank/ROOT/initial@auto-‘date +%Y-%m-%d‘
# zfs list -t snapshot
```

```
NAME                              USED  AVAIL  REFER  MOUNTPOINT
tank/ROOT/initial@auto-2016-09-13  136M      -  16.8G  -
tank/ROOT/initial@auto-2016-09-19  304K      -  16.9G  -
...
```

zpool subcommands

add

attach

clear

create

destroy

detach

export

get

history

import

iostat

list

offline

online

remove

replace

scrub

set

status

upgrade

```
# zpool iostat -v
```

```
              capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
tank        21.7G  56.3G      1      0  39.8K  3.58K
  ada0p2    21.7G  56.3G      1      0  39.8K  3.58K
----------  -----  -----  -----  -----  -----  -----
```

ZFS storage pools

- zero or more hot spares allocated at pool-creation time, or later:

  ```
  # zpool create tank mirror c0t0d0 c0t1d0 spare c0t2d0
                                  ⇐ two disks in pool with one spare
  # zpool replace tank c0t0d0 c0t3d0   ⇐ replace bad disk c0t0d0
  # zpool remove tank c0t2d0                   ⇐ remove hot spare
  ```

- hot spares can be shared across multiple pools
- easy expansion: `zpool add pool vdev`
- disk-size agnostic [though best if pool members are identical]
- disk-vendor agnostic
- pools can grow, but **cannot shrink**
- optional quotas provide additional level of usage control within a pool
- quotas can oversubscribe pool storage
- quotas can grow or shrink:

  ```
  # zfs set quota=50G saspool01/students
  ```

ZFS data replication and recovery

- none (no media-failure protection)
- stripe over *n* disks (fast, but no media-failure protection)
- mirror: recover from failure of 1 of 2 disks
- triple-mirror: recover from failure of 2 of 3 disks
- RAID Z1: recover from failure of 1 of 4 or more disks
- RAID Z2: recover from failure of 2 of 9 or more disks
- RAID Z3: recover from failure of 3 of many disks

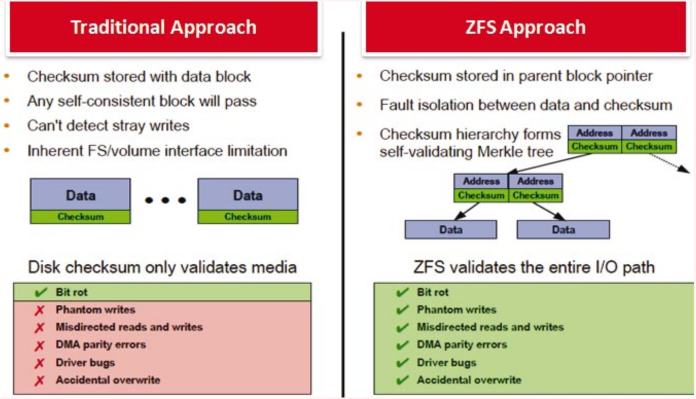
Recoverable data errors result in replacement of the erroneous block, making ZFS **self healing**.

ZFS performance and reliability

- optional compression with LZJB, LZ4, GZIP, GZIP-2, or GZIP-9 algorithms
- compression may increase performance by reducing data transfer, as long as enough spare CPU cycles are available
- copy-on-write policy means that existing data blocks are never overwritten: once the new blocks are safely in place, old blocks are freed for re-use if they are not in a snapshot
- supports *n*-way mirrors: a mirror of n disks can lose up to $n - 1$ disks before data loss
- supports striping and RAID-Z[1-3]
- internal per-block checksums [no hardware RAID needed or desirable: JBOD is good enough because ZFS is better than RAID]
- ZFS is being optimized for SSDs, which suffer from severe **wear limits** that ultimately reduce disk and pool capacity

ZFS checksums

Unlike most other filesystems, each data and metadata block of a ZFS filesystem has a SHA-256 checksum stored in the *pointer to the block*, not in the block itself, so less subject to corruption.



Traditional Approach

- 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 and writes
- ✗ DMA parity errors
- ✗ Driver bugs
- ✗ Accidental overwrite

ZFS Approach

- 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

- ✓ Bit rot
- ✓ Phantom writes
- ✓ Misdirected reads and writes
- ✓ DMA parity errors
- ✓ Driver bugs
- ✓ Accidental overwrite

[From **Architectural Overview of the Oracle ZFS Storage Appliance**]

Checksums on new blocks are recalculated, not copied, and errors can be corrected if there is sufficient redundancy (mirror or RAID-Zn replication).

ZFS snapshots

- fast: one or two seconds, independent of filesystem size
- unlimited number of snapshots
- snapshots are **read-only**
- snapshots are user visible [e.g., `/.zfs/snapshot/auto-2016-10-11/home/jones/mail`]
- `/.zfs` normally hidden from directory listing commands [management configurable]
- disk blocks captured in a snapshot are in use until snapshot is destroyed
- removing recent large files on a disk-full condition may free **no space at all**: instead, need to **remove oldest snapshots**
- a snapshot can be **clone**d to create a new **writable** filesystem

ZFS scrub, resilver, and ZIL

- **scrub** is root-initiated dynamic consistency check, run in **background** on **mounted** live filesystem, so no denial-of-service as in traditional $fsck$
- **resilver** is automatic dynamic consistency restoration run after a disk or network failure, or slowdown of one or more mirrors
- **ZIL** is *ZFS Intent Log*: a journal of metadata commits; it can optionally be kept in a different filesystem, perhaps on solid-state drives (SSDs)

ZFS scrub example

```
# mount | grep zfs
tank/ROOT/initial on / (zfs, local, noatime, nfsv4acls)
...
# zpool scrub tank
# zpool status
  pool: tank
 state: ONLINE
  scan: scrub in progress since Tue Oct 11 18:07:36 2016
        14.0M scanned out of 21.7G at 895K/s, 7h2m to go
        0 repaired, 0.06% done
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          ada0p2    ONLINE       0     0     0
errors: No known data errors
```

ZFS boot environments

- on some O/Ses with ZFS, critical system updates are done in a new *boot environment* that is not visible until selected at the next boot
- if a problem appears in the new environment, just reboot into most stable recent boot environment
- analogous to *grub*, *lilo*, *silo*, or other boot loader, offer of multiple kernels at boot time, but includes much more than just the kernel

ZFS inter-site mirrors

- *n*-way live mirroring
- we use 8Gb/s FibreChannel connect to ZFS mirror in another campus building
- read requests can be served by any mirror
- if one mirror goes away, file serving continues transparently from another mirror
- when lost mirror comes back, a **resilver** operation eventually makes all mirrors consistent [but may take hours or days]

ZFS send, receive, and live migration

For convenient filesystem backup:

- initial *zfs send* of a ZFS filesystem snapshot to a remote machine running *zfs receive* duplicates filesystem (assuming compatible ZFS feature levels)
- remote machine has working [but out-of-date] copy of original filesystem: probably okay for HTTP and FTP services, library catalogs, and other reasonably stable databases
- subsequent *zfs send* transfers only a snapshot that is usually much smaller than original filesystem
- *zfs receive* can pull back a filesystem from a remote machine to repopulate a replaced or repaired local filesystem

Can **migrate entire live filesystem** to new storage technology with replacement of old disks by bigger new disks using **resilver** feature.

ZFS as root filesystem

- Some O/Ses [Solaris, ghostbsd, PC-BSD, and TrueOS] can boot from ZFS filesystem
- Other O/Ses [Debian, FreeBSD, Ubuntu] need a small native UFS [or FFS, JFS, Reiser, XFS, . . .] filesystem for /boot partition, with remaining data on ZFS
- Several Linux distributions have optional ZFS support [we run it on CentOS, Debian, Fedora, Red Hat, and Ubuntu]
- Fully-bootable ZFS coming on Debian and Debian-like [ElementaryOS, Kali, Knoppix, Mint, Salix, Ubuntu, and others] GNU/Linux systems

ZFS underneath virtual machines

- SmartOS is a minimal OpenSolaris-based system with zones, ZFS, and a port of Linux KVM. SmartOS provides an alternative to Hyper-V, QEMU, VirtualBox, VMware, Xen, and other virtualization environments [News: Samsung bought out Joyent, maker of SmartOS, in June 2016]
- VM filesystem backup and snapshot really requires communication between virtualization layer and VM O/S or database, but only a few O/Ses have the needed kernel drivers to support that
- Without such synchronization, a restored backup or snapshot may well be unusable in a VM because of filesystem inconsistencies

ZFS vs. GNU/Linux btrfs

Compared to GNU/Linux btrfs, ZFS

- is developed and supported on multiple O/Ses, and thus not tied to one O/S kernel flavor
- can be imported to, and exported from, other O/Ses with *zpool import* and *zpool export*
- is capable of much larger filesystem capacity
- is more mature and stable
- has more features, with deduplication, compression, encryption [not yet in OpenZFS], . . .
- snapshots appear to take much less space than in **btrfs**
- seems to reclaim disk space much faster from freed snapshots [personal observation]

ZFS in network-attached storage appliances

Even if you cannot, or will not, manage ZFS on your fileserver, you can buy turn-key appliances that contain ZFS:

- Dell Compellent NAS
- EON ZFS Storage
- iXsystems FreeNAS and TrueNAS
- Oracle ZFS Storage Appliance [includes ARC — Adaptive Replacement Cache (DRAM level-1 cache), plus L2ARC (SSD level-2 cache, and ZIL in SSD)]
- Polywell PolyStor
- QNAP ES (Enterprise Storage) NAS
- Tegile all-flash and hybrid-flash arrays
- Zeta Storage Systems
- others?

Other filesystems with snapshots

- Apple Mac OS X **Time Machine** [incremental backups to remote storage with time-slice views]
- DragonFlyBSD **hammer** [automatic snapshots of active files]

```
% undo -i myfile
myfile: ITERATE ENTIRE HISTORY
        0x0000000102b96fa0 18-Aug-2016 09:57:49
        0x000000010e6621f0 27-Sep-2016 17:42:13 file-deleted
        0x000000010e662310 27-Sep-2016 17:52:07 inode-change
        0x00000001128d8110 08-Oct-2016 09:13:47
% undo -u myfile
% undo -u -t0x000000010e662310 myfile
% ls -log myfile*
-rwxr-xr-x 1 132026 Oct  8 09:13 myfile
-rw-rw-r-- 1 130023 Oct 12 06:26 myfile.undo.0000
```

- GNU/Linux **btrfs** volume-based snapshots [read-only, or writable]
- NetApp network attached storage (NAS) with proprietary **WAFL** filesystem with up to 255 snapshots per volume, visible in special hidden subdirectory `.snapshot` of each directory
- others?

ZFS from a developer's viewpoint

See interview with Richard Yao on *BSD Now TV* program *Episode 157: ZFS, The "Universal" File-system*:

`https://www.bsdnow.tv/episodes/`

Yao says that there *are* ways to lose your *entire ZFS filesystem*, even though they are rare [we've never seen such a loss].
All filesystems need to be backed up, and preferably, redundantly!
See also the MeetBSD 2016 conference video **OpenZFS: History of ZFS** by ZFS architect Matt Ahrens:

`https://www.youtube.com/watch?v=Hz7CEI8LwSI`

Our ZFS development wishlist

- shrinkable storage pools
- automatic drive capacity rebalancing in background after a pool is grown [or, in the future, shrunk]
- view into pool disks: free and used space, error counts, I/O stats, . . .
- better utilization of pool of disks of mixed sizes [e.g., from technology improvements over time]
- contiguous files [for maximal streaming performance]
- preallocated files [to prevent run-time out-of-space condition]; partly available by `# zfs set reservation=nnnn`
- traditional Unix access controls are based on 3 local categories: *user*, *group*, and *other*: need more, such as *client*, *customer*, and *world*
- NetApp WAFL-like `.snapshot` subdirectory of each directory
- quality-of-service (QoS) guarantee for ZFS I/O
- **platform-independent GUI** for visual control of disks, pools, mirroring, RAIDing, and striping, with visual warnings for excess use or errors [partially available with Sun StorAid or Oracle ZFS Appliance]

Sample ZFS I/O statistics

```
# zpool iostat -v

                 capacity     operations    bandwidth
pool          alloc   free   read  write   read  write
------------  -----  -----  -----  -----  -----  -----
pool01        9.53T  1.34T     33     19  3.65M  2.35M
  raidz1      2.38T   342G      8      4   934K   602K
    c0t2d0        -      -      4      1   184K   122K
    c1t2d0        -      -      4      1   184K   122K
    c2t2d0        -      -      4      1   184K   122K
    c3t2d0        -      -      4      1   184K   122K
    c4t2d0        -      -      4      1   184K   122K
    c4t7d0        -      -     14     16   320K   205K
...
------------  -----  -----  -----  -----  -----  -----
rpool          112G   352G      3      6  57.1K  27.1K
  mirror       112G   352G      3      6  57.1K  27.1K
    c3t0d0s0      -      -      1      3  52.3K  27.2K
    c3t4d0s0      -      -      1      3  52.3K  27.2K
------------  -----  -----  -----  -----  -----  -----
```

Books on ZFS

- Sun, **Solaris ZFS administration guide** (2008), ISBN 0-595-35252-9
- Scott Watanabe, **Solaris 10 ZFS essentials** (2010), ISBN 0-13-700010-3
- Nicholas A. Solter, Jerry Jelinek, and David Miner, **OpenSolaris Bible** (2009), ISBN 0-470-38548-0
- Thomas W. Doeppner, **Operating Systems In Depth: Design and Programming** (2011), ISBN 0-471-68723-5
- Marshall Kirk McKusick and George V. Neville-Neil, **The Design and Implementation of the FreeBSD Operating System**, 2nd edition (2014), ISBN 0-321-96897-2
- Michael W. Lucas and Allan Jude, **FreeBSD Mastery: ZFS** (2015), ISBN 0-692-45235-4
- Allan Jude and Michael W. Lucas, **FreeBSD Mastery: Advanced ZFS** (2016), ISBN 0-692-68868-4
- Oracle, **Architectural Overview of the Oracle ZFS Storage Appliance** (2016).

Web resources on ZFS

http://learnxinyminutes.com/docs/zfs/

http://open-zfs.org/wiki/Performance_tuning

http://wiki.freebsd.org/ZFSTuningGuide

http://www.bsdnow.tv/tutorials/zfs

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/zfs.html

http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

http://www.solarisinternals.com/wiki/index.php/ZFS_Configuration_Guide

http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide