ZFS

Zetabyte FileSystem

The Last Word In File Systems

chiachunt

Are you tired of…

- ❑ Disk Partitioning (Slice,label….)
- ❑ Physical volume limitation
- ❑ Long FSCK time
- ❑ Silent corruption on your disk
- ❑ waiting when moving large files

…etc

Try ZFS!

Outline

- Z File System Introduction
- Using ZFS
  - zpool
  - zfs
- Reference
  - http://hub.opensolaris.org/bin/view/Community+Group+zfs/
    - http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/zfslast.pdf
  - http://en.wikipedia.org/wiki/ZFS

ZFS Intro

- ❑ Storage Pools
  - Constructed of files, partitions, or entire disks
    - ➢ Does for storage what VM did for memory
  - stripe, mirror(RAID1), raidz(RAID5), raidz2, raidz3
    - ➢ Hot spares
- ❑ Capacity
  - 128-bit file system
- ❑ Max File Size
  - 16 EB (KB, MB, GB, TB, PB, EB)
- ❑ Max Volume Size
  - 16 EB

Break the limitation of traditional File System!

ZFS Intro

- Variable block size
  - Data compression (CPU-bound vs. I/O-bound)
- Adaptive endianness
  - Sparc(Big-endian) & others(Little-endian)

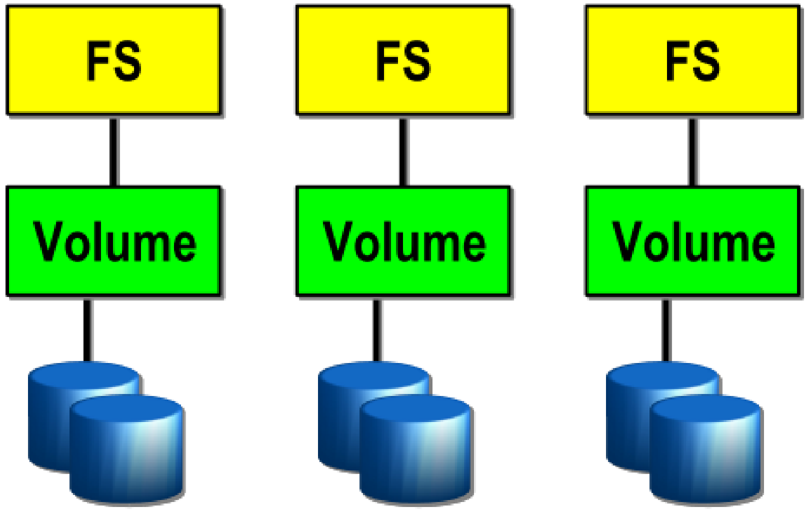
FS/Volume model vs. ZFS

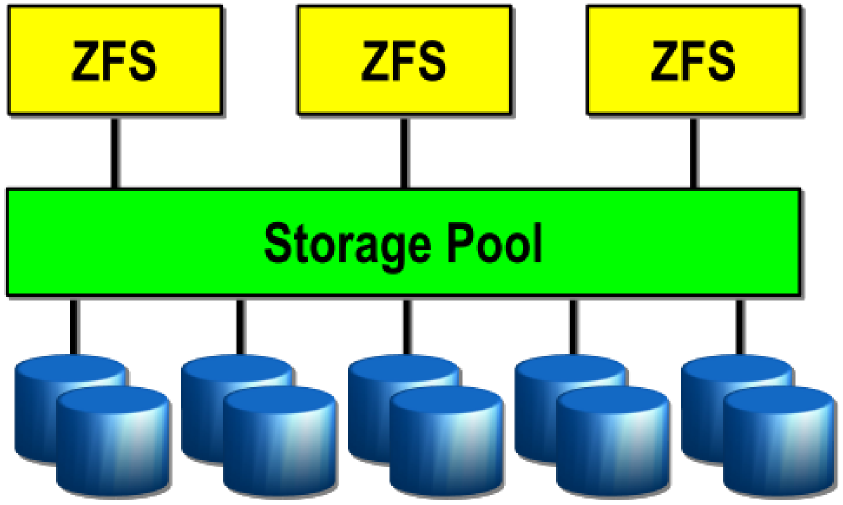
Traditional Volumes

- 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





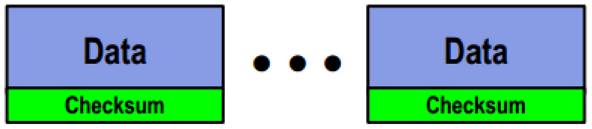
ZFS Intro (Cont.)

❑ Data Integrity
- Checksums
- Online "scrub"
  - ➢ fsck is offline and only check metadata
  - ➢ scrub once per week for cheap disks or per month for enterprise disks

No FSCK and silent corruption!

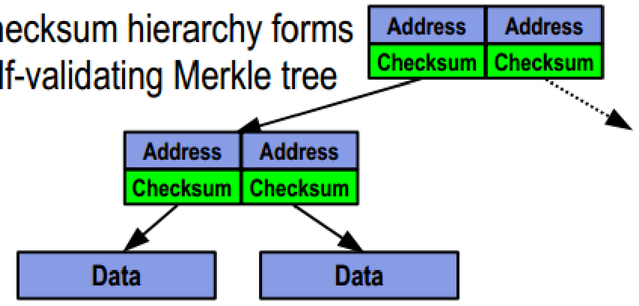
Disk Block Checksums

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



ZFS Data Authentication

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

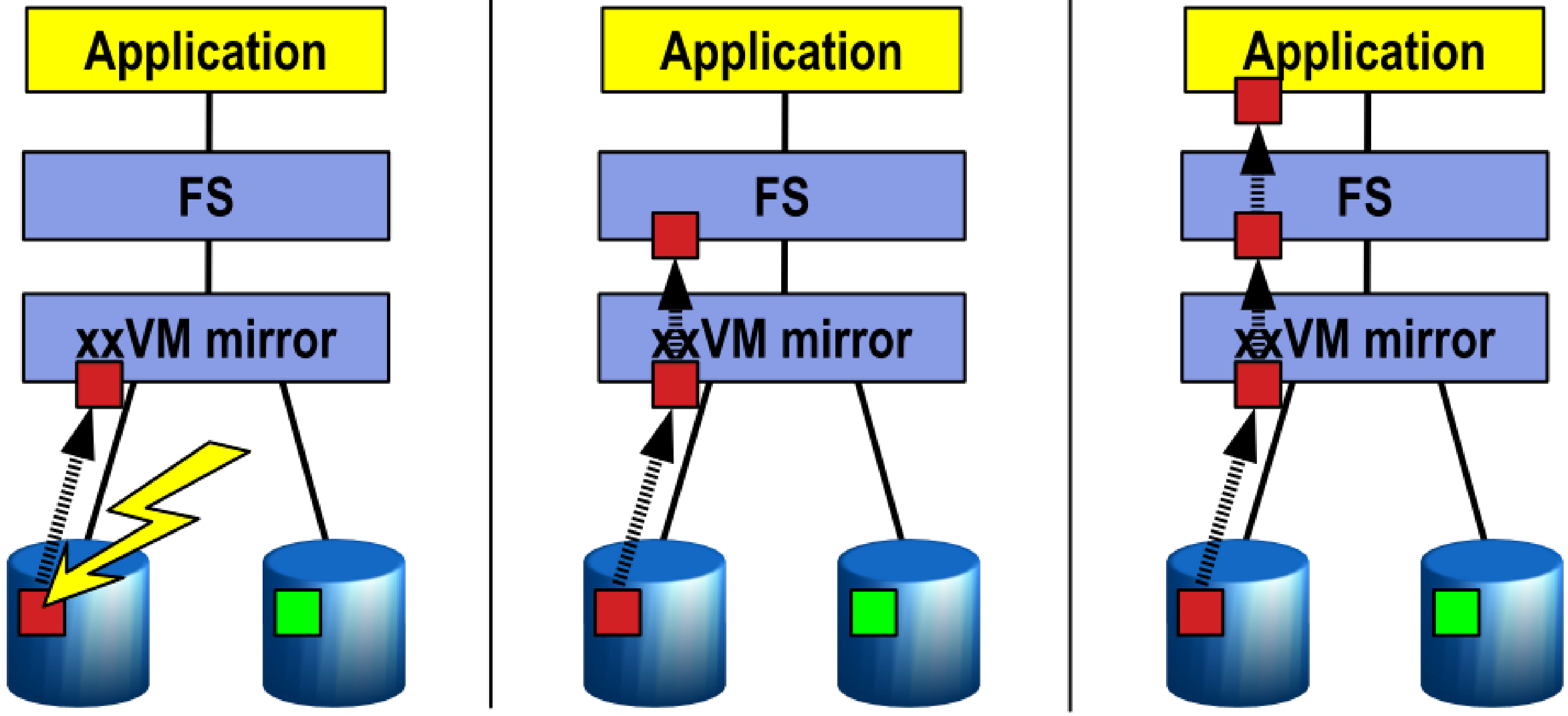


Traditional Mirroring

1. Application issues a read. Mirror reads the first disk, which has a corrupt block. It can't tell.

2. Volume manager passes bad block up to filesystem. If it's a metadata block, the filesystem panics. If not...

3. Filesystem returns bad data to the application.

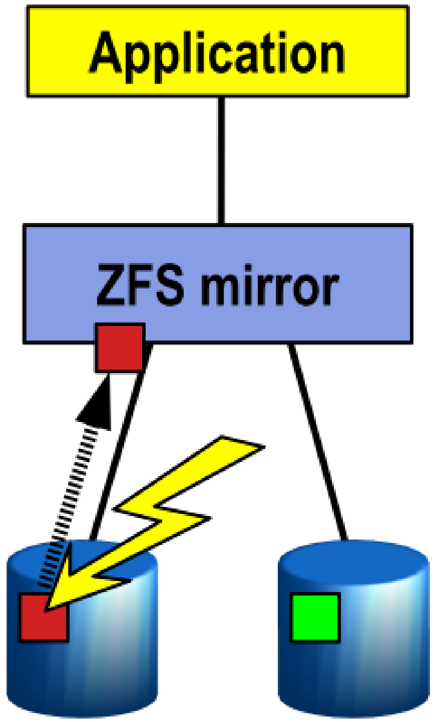


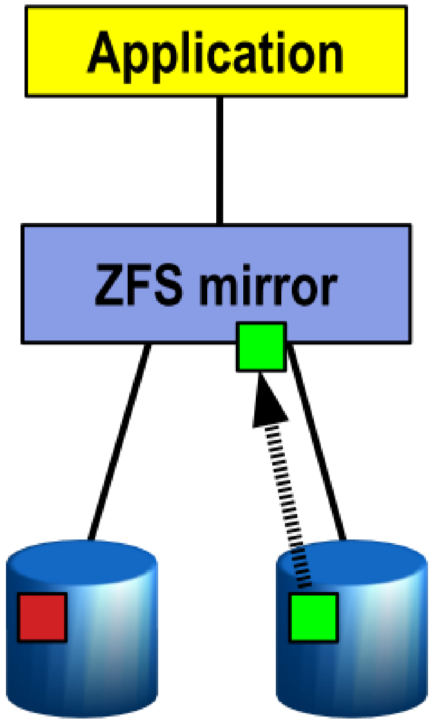
Self-Healing in ZFS

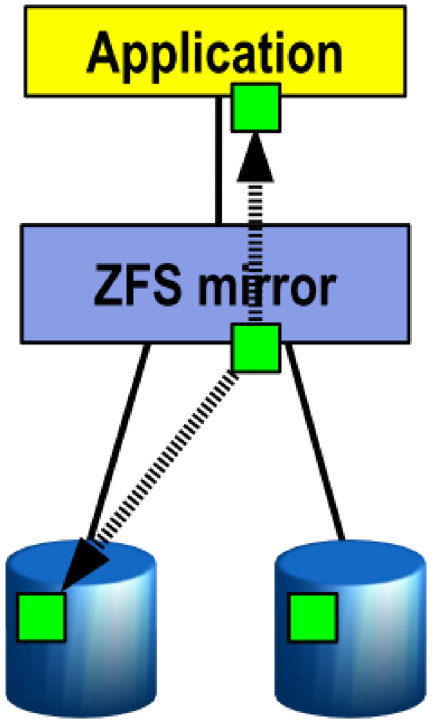
1. Application issues a read. ZFS mirror tries the first disk. Checksum reveals that the block is corrupt on disk.

2. ZFS tries the second disk. Checksum indicates that the block is good.

3. ZFS returns known good data to the application and repairs the damaged block.







ZFS Intro (Cont.)

- Dynamic striping
  - Across all devices to maximize throughput
  - What if I only have one disk….
- Copy-on-write
  - Modified data is in a new block

Improve throughput and efficiency

ZFS Intro – Other Features

- Snapshots & Clones & Rollbacks
  - Fast creation
  - Space efficiency
  - Clones are writeable snapshots
- Compression
  - lzjb, gzip-*
- Encryption
  - ZFS Pool Version 30

ZFS – Platforms

- Solaris 10 / 11
- OpenSolaris / OpenIndiana
- FreeBSD
  - 8.2-R: v15
  - 8.2-S, 9.0-, 10.0-C: v28
- FreeNAS
- GNU/kFreeBSD
- NetBSD
- Mac X OS
- Linux / Linux FUSE / Kernel Module

Using ZFS

- zpool
  - Maintain relations between harddisk and ZFS pool
- zfs
  - Manage ZFS and configuration setting
- In /etc/rc.conf
  - zfs_enable="YES“

```
# /etc/rc.d/zfs start
```

zpool

- zpool create tank
  - ad0 ad1
  - mirror ad0 ad1
  - raidz ad0 ad1 ad2
- zpool destroy tank
- zpool add/attach/detach/online/offline
- zpool clear/replace
- zpool list/status
- zpool export/import
- zpool upgrade -v/-a
- zpool get/set …
  - get all

zfs – (1)

- zfs create
  - tank/fs0
- zfs destroy
  - tank/fs0
- zfs snapshot
  - tank/fs0@today
  - tank/fs0/.zfs/snapshot/today/
- zfs clone
- zfs rollback
- zfs list

zfs – (2)

- ❑ zfs mount/unmount
- ❑ zfs upgrade -v/-a
- ❑ zfs send/receive
- ❑ zfs allow/unallow
- ❑ zfs hold/holds/release
- ❑ zfs diff
- ❑ zfs jail/unjail

Under these attracting features

There must be a price…

ZFS can be a monster which eat up all your memory

So watch out your hardware limit

or it will CRASH!

Ten Ways To Improve ZFS Performance

- ❑ 1: Add Enough RAM
- ❑ 2: Add More RAM
- ❑ 3: Boost Deduplication Performance With Even More RAM
- ❑ 4: Use SSDs to Improve Read Performance
- ❑ 5: Use SSDs to Improve Write Performance
- ❑ 6: Use Mirroring
- ❑ 7: Add More Disks
- ❑ 8: Leave Enough Free Space
- ❑ 9: Hire An Expert
- ❑ 10: Be An Evil Tuner - But Know What You Do

http://constantin.glez.de/blog/2010/04/ten-ways-easily-improve-oracle-solaris-zfs-filesystem-performance

Tuning for production use

- Users of the i386™ architecture
  - add in kernel configuration file, rebuild their kernel, and reboot
    - options KVA_PAGES=512
- There is one example of ZFS running nicely on a laptop with 768 MB of physical RAM with the following settings in /boot/loader.conf:
  - vm.kmem_size="330M"
  - vm.kmem_size_max="330M"
  - vfs.zfs.arc_max="40M"
  - vfs.zfs.vdev.cache.size="5M“
- http://wiki.freebsd.org/ZFSTuningGuide

Further Reading

- ❑ http://www.freebsd.org/doc/handbook/filesystems-zfs.html
  - zpool(1M)、zfs(1M)
- ❑ http://wiki.freebsd.org/ZFS
  - Live demos!!!!
- ❑ http://wiki.freebsd.org/ZFSQuickStartGuide
- ❑ http://wiki.freebsd.org/RootOnZFS
  - ZFS-only FreeBSD

Appendix – Create zpool using files

- ❑ Prepare BIG blank files
  - dd if=/dev/zero of=/zfs/device/disk1 bs=1m count=1024
  - …
- ❑ Create zpool
  - zpool create filetank mirror /zfs/device/disk1 …