Porting ZFS[1] file system to FreeBSD[2]

Paweł Jakub Dawidek
<pjd@FreeBSD.org>

[1] last word in file systems
[2] last word in operating systems

Do you plan to use ZFS in FreeBSD 7?

Have you already tried FreeBSD/ZFS?

Do you use FreeBSD/ZFS in production?

The beginning...

- ZFS released by SUN under CDDL license
- available in Solaris / OpenSolaris only
- ongoing Linux port for FUSE framework (userland); started as SoC project
- ongoing top secret port for MacOS X

Features...

· ZFS has many very interesting features, which makes it one of the most wanted file systems

Features...

- dynamic stripping – use the entire bandwidth available,
- RAID-Z (RAID-5 without "write hole" (more like RAID-3 actually)),
- RAID-1,
- 128 bits (POSIX limits FS to 64 bits)... (think about 65 bits)

Features...

- pooled storage
  - no more volumes/partitions
  - does for storage what VM did for memory
- copy-on-write model
- transactional operation
  - always consistent on disk
  - no fsck, no journaling
- intelligent synchronization (resilvering)
  - synchronize only valid data

Features...

- snapshots
  - very cheap, because of COW model
- clones
  - writtable snapshots
- snapshot rollback
  - very handy "undo" operation
- end-to-end data integrity
  - detects and corrects silent data corruption caused by any defect in disk, cable, controller, driver or firmware

Features...

- built-in compression
  - lzjb, gzip
- self-healing
  - return good data and fix corrupted data
- endian-independent
  - always write in native endianess
- simplified administration
- per-filesystem encryption
  - soon

FS/Volume model vs. ZFS

Traditional Volumes
- abstraction: virtual disk
- volume/partition for each FS
- grow/shrink by hand
- each FS has limited bandwidth
- storage is fragmented
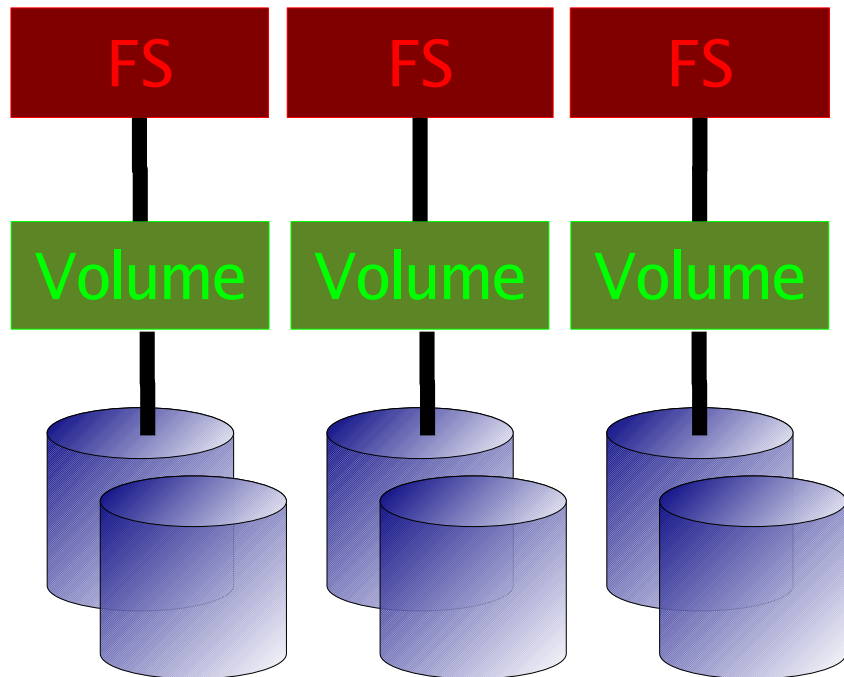
ZFS Pooled Storage
- abstraction: malloc/free
- no partitions to manage
- grow/shrink automatically
- all bandwidth always available
- all storage in the pool is shared
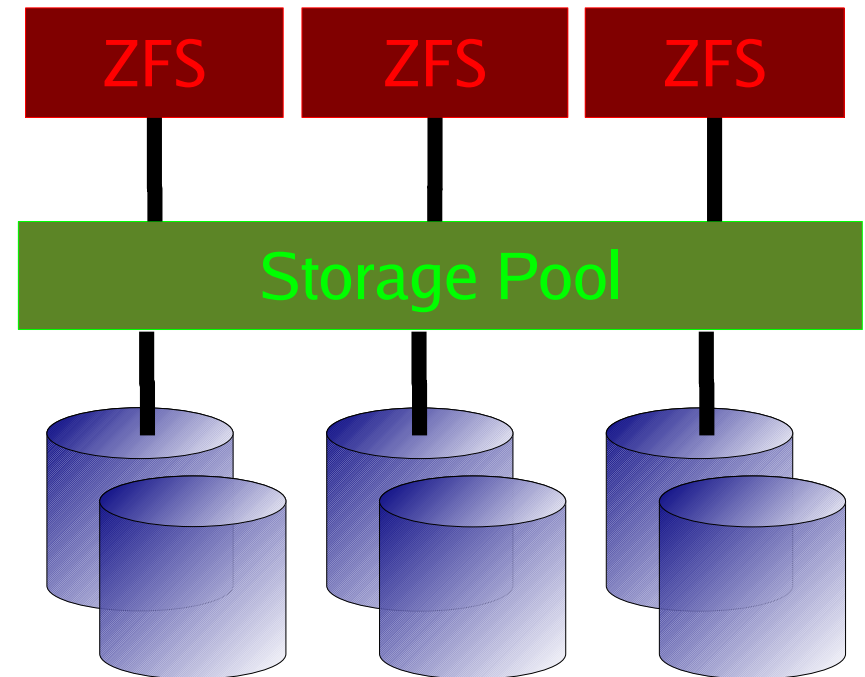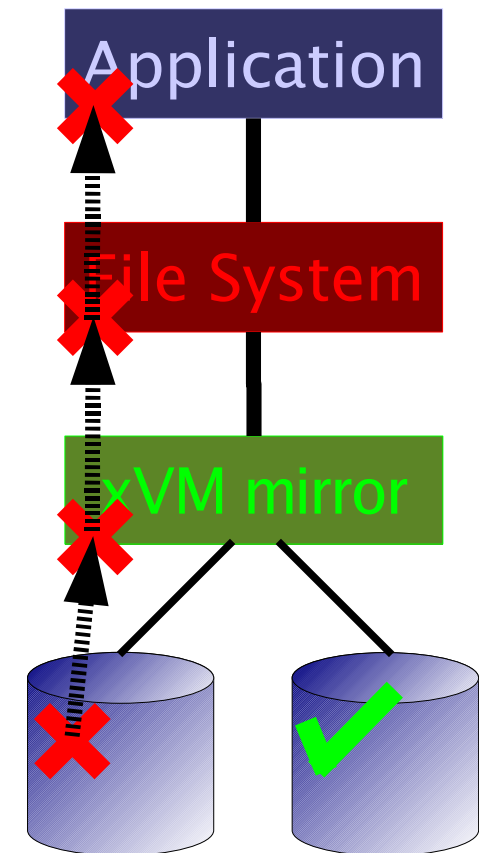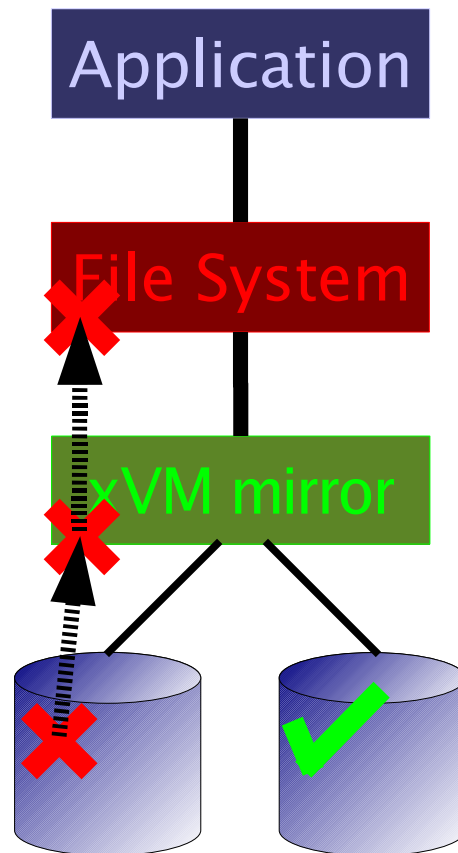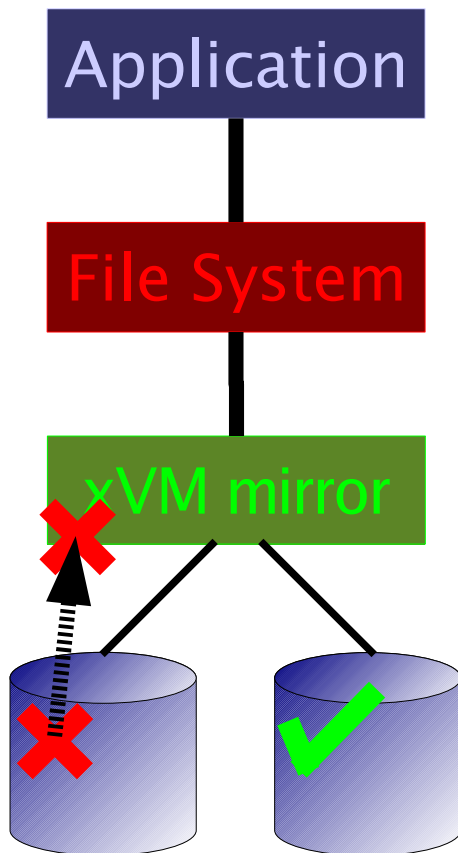
ZFS Self-Healing

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 the bad block to file system. If it's a metadata block, the system panics. If not...

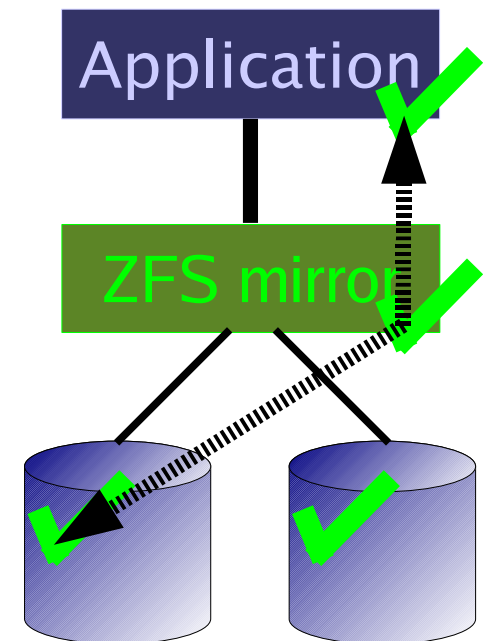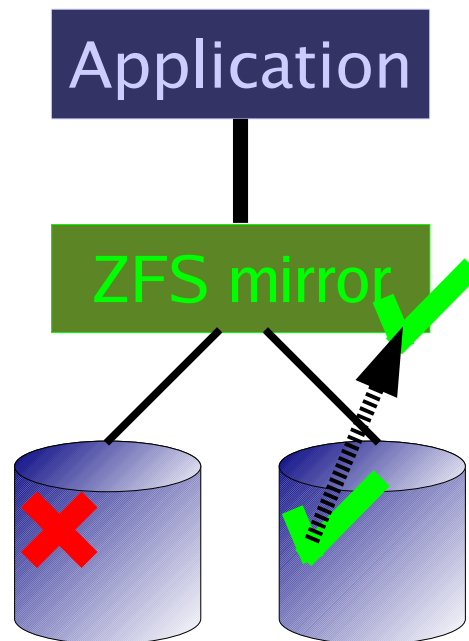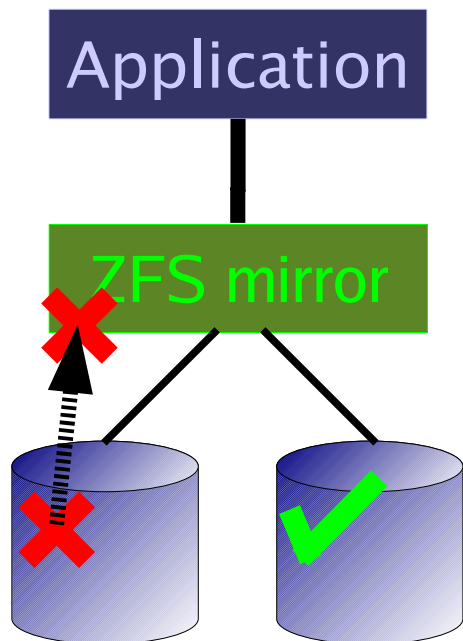3. File system returns bad data to the application...

Self-Healing data 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 good data to the application and repairs the damaged block.

Porting...

· very portable code (started to work after 10 days (and nights) of porting)

· few ugly Solaris-specific details

· few ugly FreeBSD-specific details (VFS, buffer cache)

· ZPL was hell (ZFS POSIX layer); yes, this is the thing which VFS talks to

Solaris compatibility layer

contrib/opensolaris/ - userland code taken from OpenSolaris
   used by ZFS (ZFS control utilities, libraries, test tools)
compat/opensolaris/ - userland API compatibility layer
   (Solaris-specific functions missing in FreeBSD)
cddl/ - Makefiles used to build userland libraries and utilities
sys/contrib/opensolaris/ - kernel code taken from OpenSolaris
   used by ZFS
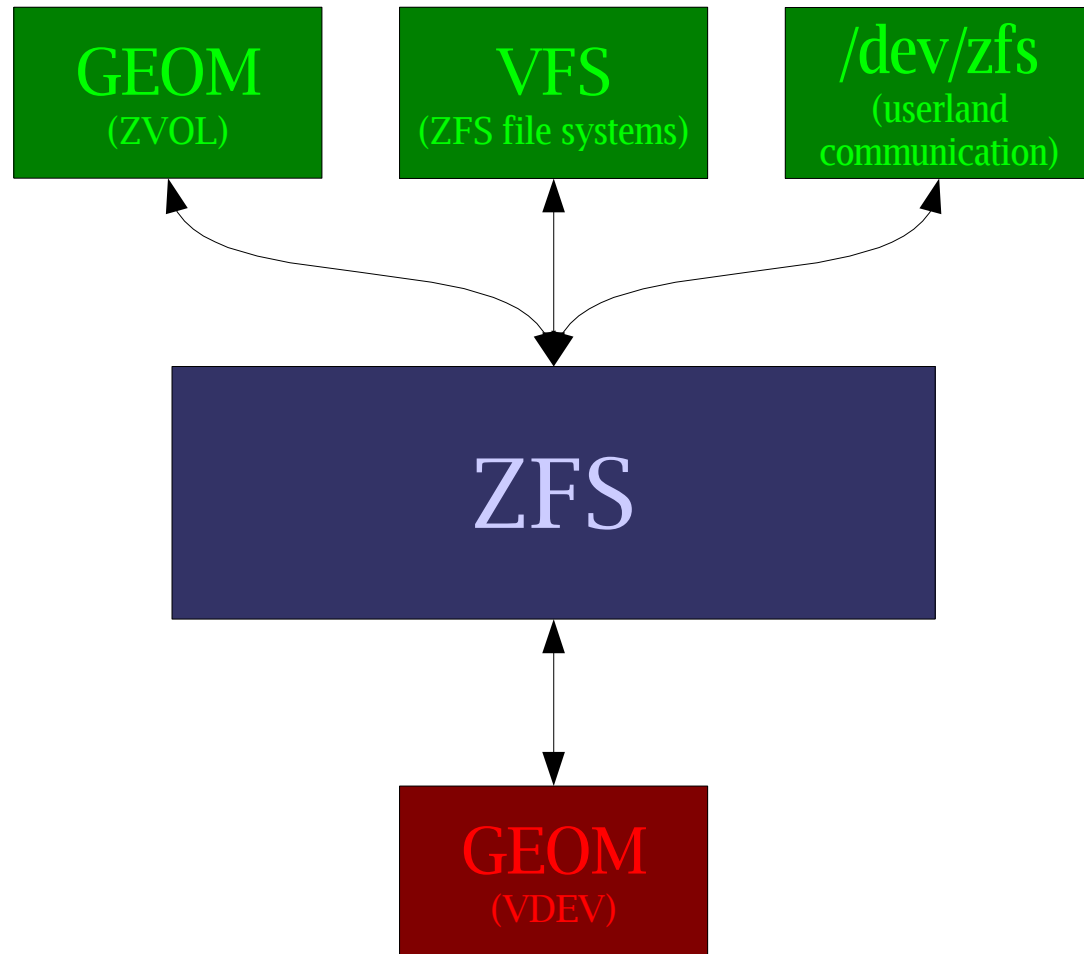sys/compat/opensolaris/ - kernel API compatibility layer
sys/modules/zfs/ - Makefile for building ZFS kernel module
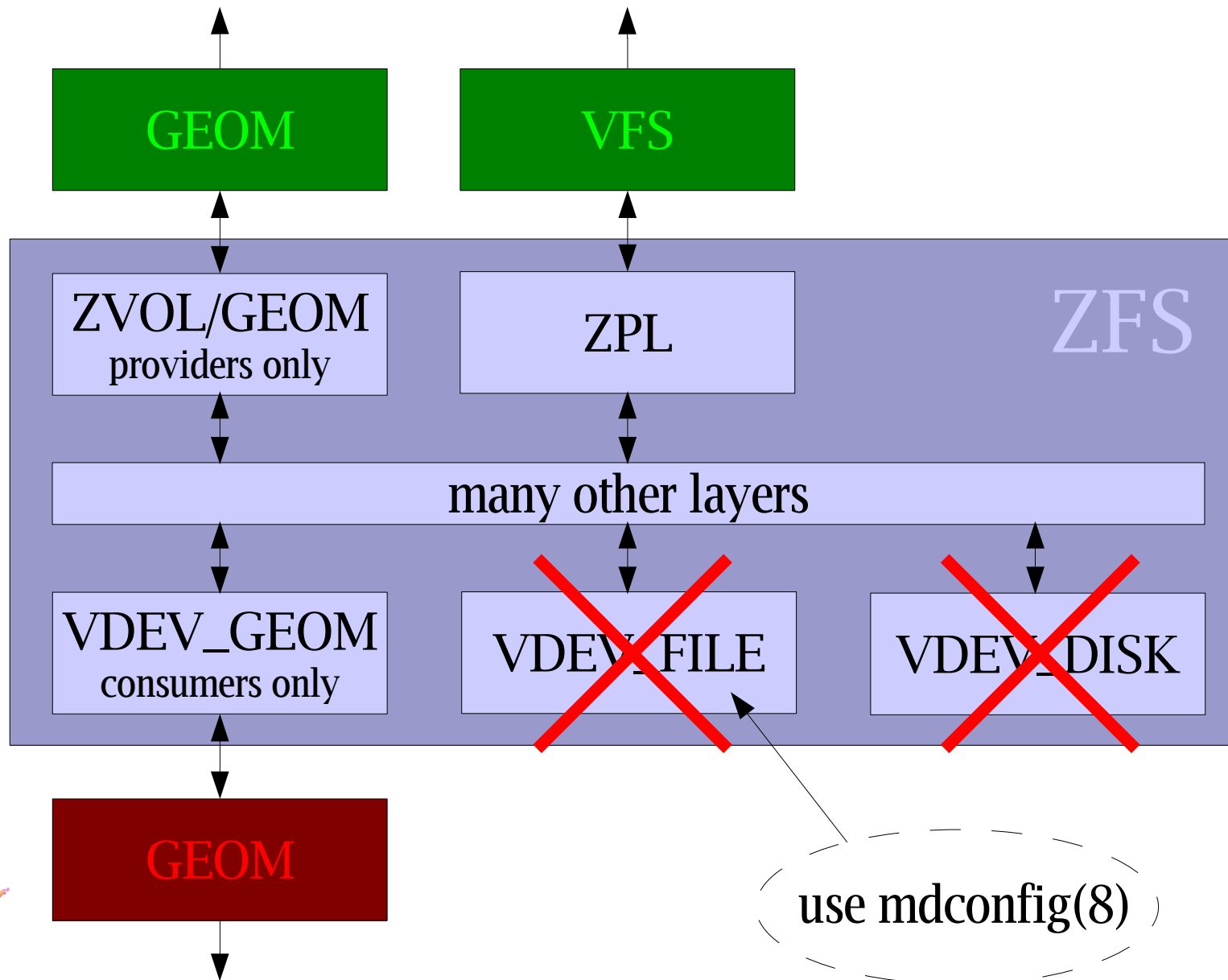
ZFS connection points in the kernel

How does it look exactly...

Snapshots

- contains @ in its name:

```
# zfs list
NAME                    USED    AVAIL   REFER   MOUNTPOINT
tank                    50,4M   73,3G   50,3M   /tank
tank@monday             0       -       50,3M   -
tank@tuesday            0       -       50,3M   -
tank/freebsd            24,5K   73,3G   24,5K   /tank/freebsd
tank/freebsd@tuesday    0       -       24,5K   -
```

- mounted on first access under /mountpoint/.zfs/snapshot/<name>
- hard to NFS-export
  - separate file systems have to be visible when its parent is NFS-mounted

NFS is easy

```
# mountd /etc/exports /etc/zfs/exports
# zfs set sharenfs=ro,maproot=0,network=192.168.0.0,mask=255.255.0.0 tank
# cat /etc/zfs/exports
# !!! DO NOT EDIT THIS FILE MANUALLY !!!

/tank    -ro -maproot=0 -network=192.168.0.0 -mask=255.255.0.0
/tank/freebsd   -ro -maproot=0 -network=192.168.0.0 -mask=255.255.0.0
```

· we translate options to exports(5) format and SIGHUP mountd(8) daemon

Missing bits in FreeBSD needed by ZFS

Sleepable mutexes

- no sleeping while holding mutex(9)
- Solaris mutexes implemented on top of sx(9) locks (performance improvements by Attilio Rao)
- condvar(9) version that operates on any locks, not only mutexes (implemented by John Baldwin)

GFS (Generic Pseudo-Filesystem)

- allows to create "virtual" objects (not stored on disk)
- in ZFS we have:
  .zfs/
  .zfs/snapshot
  .zfs/snapshot/<name>/

VPTOFH

- translates vnode to a file handle
- VFS_VPTOFH(9) replaced with VOP_VPTOFH(9) to support NFS exporting of GFS vnodes
- its just better that way – confirmed by Kirk McKusick

lseek(2) SEEK_{DATA,HOLE}

- SEEK_HOLE – returns the offset of the next hole
- SEEK_DATA – returns the offset of the next data
- helpful for backup software
- not ZFS-specific

Integration with jails

- ZFS nicely integrates with zones on Solaris, so why not to use it with FreeBSD's jails?
- pools can only be managed from outside a jail
- zfs file systems can be managed from within a jail

Integration with jails

```
main# zpool create tank mirror da0 da1
main# zfs create -o jailed=on tank/jail
main# jail hostname /jail/root 10.0.0.1 /bin/tcsh
main# zfs jail <id> tank/jail

jail# zfs create tank/jail/home
jail# zfs create tank/jail/home/pjd
jail# zfs snapshot tank/jail/home@today
```

Testing correctness

- ## ztest (libzpool)
  - "a product is only as good as its test suite"
  - runs most of the ZFS code in userland
  - probably more abuse in 20 seconds that you'd see in a lifetime
- ## fstest regression test suite
  - 3438 tests in 184 files
  - # prove -r /usr/src/tools/regression/fstest/tests
  - tests: chflags(2), chmod(2), chown(2), link(2), mkdir(2), mkfifo(2), open(2), rename(2), rmdir(2), symlink(2), truncate(2), unlink(2)
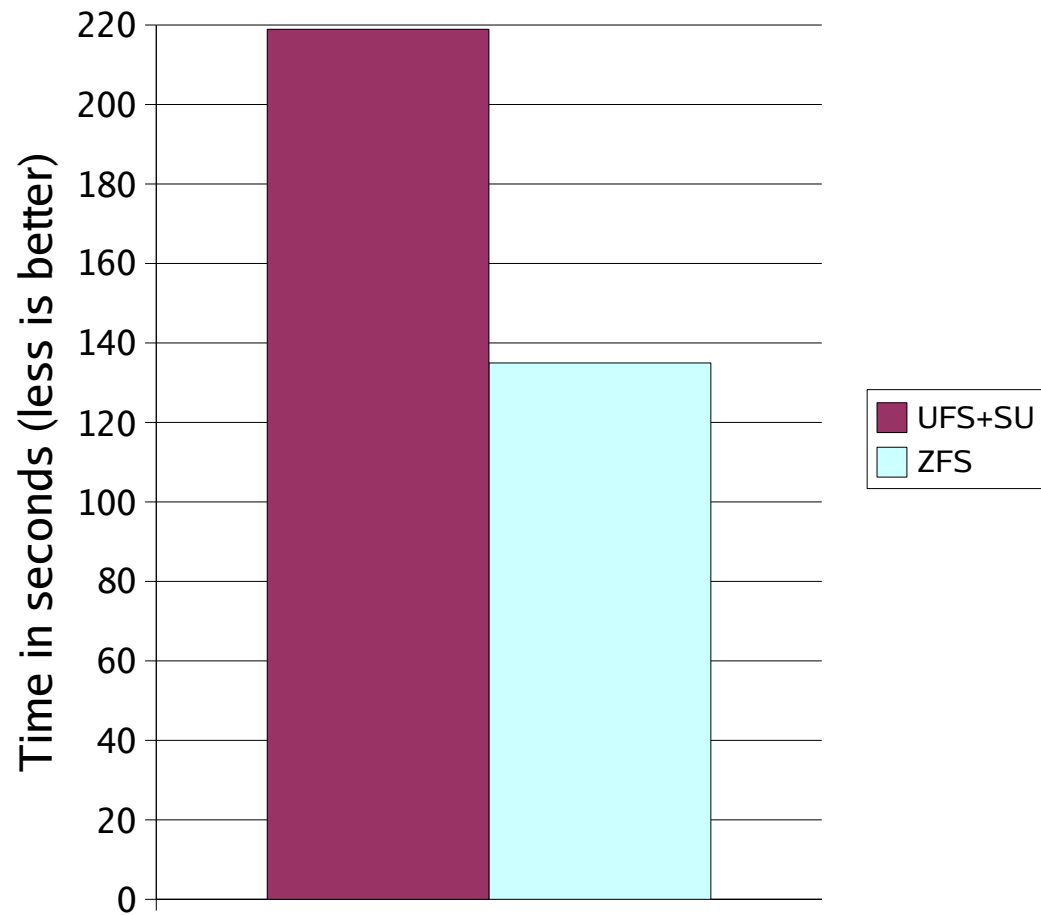
Performance
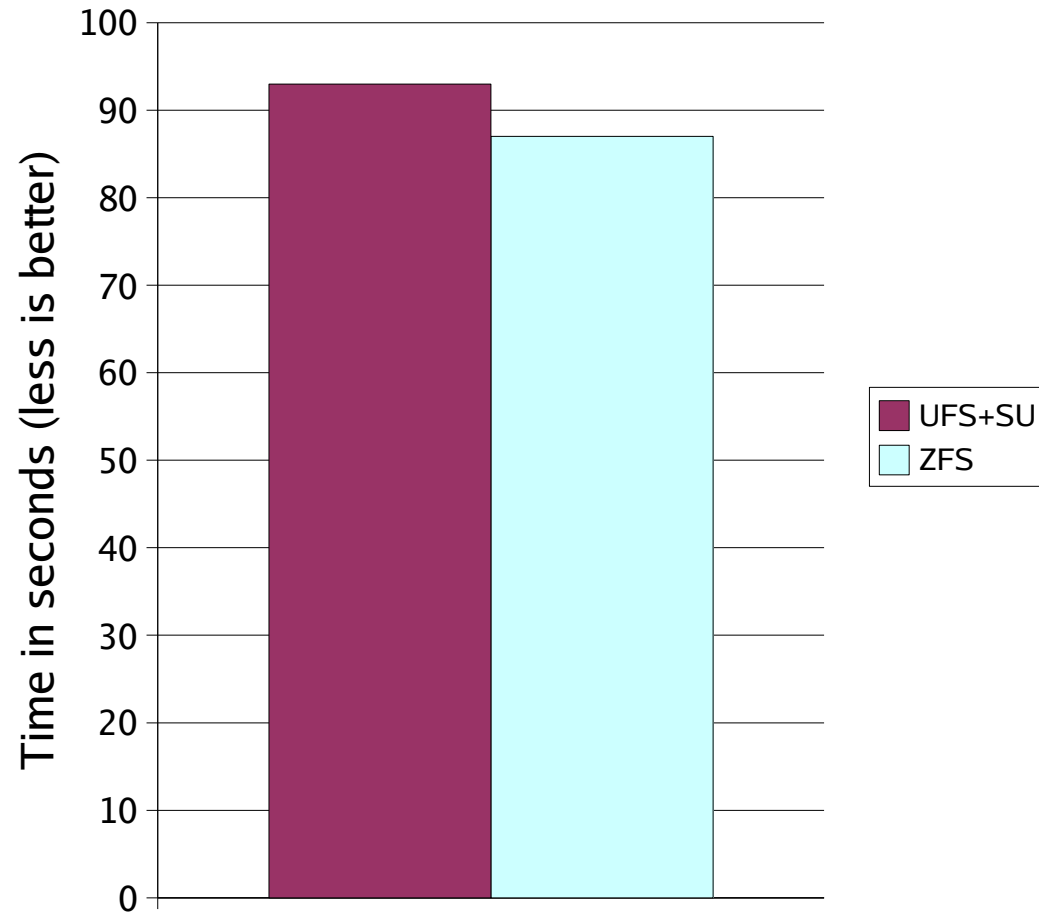
Before showing the numbers...

- a lot has been done in this area
  - the buffer cache bypass
  - new sx(9) implementation
  - namecache
  - shared vnode locking
  - mmap(2) fixes
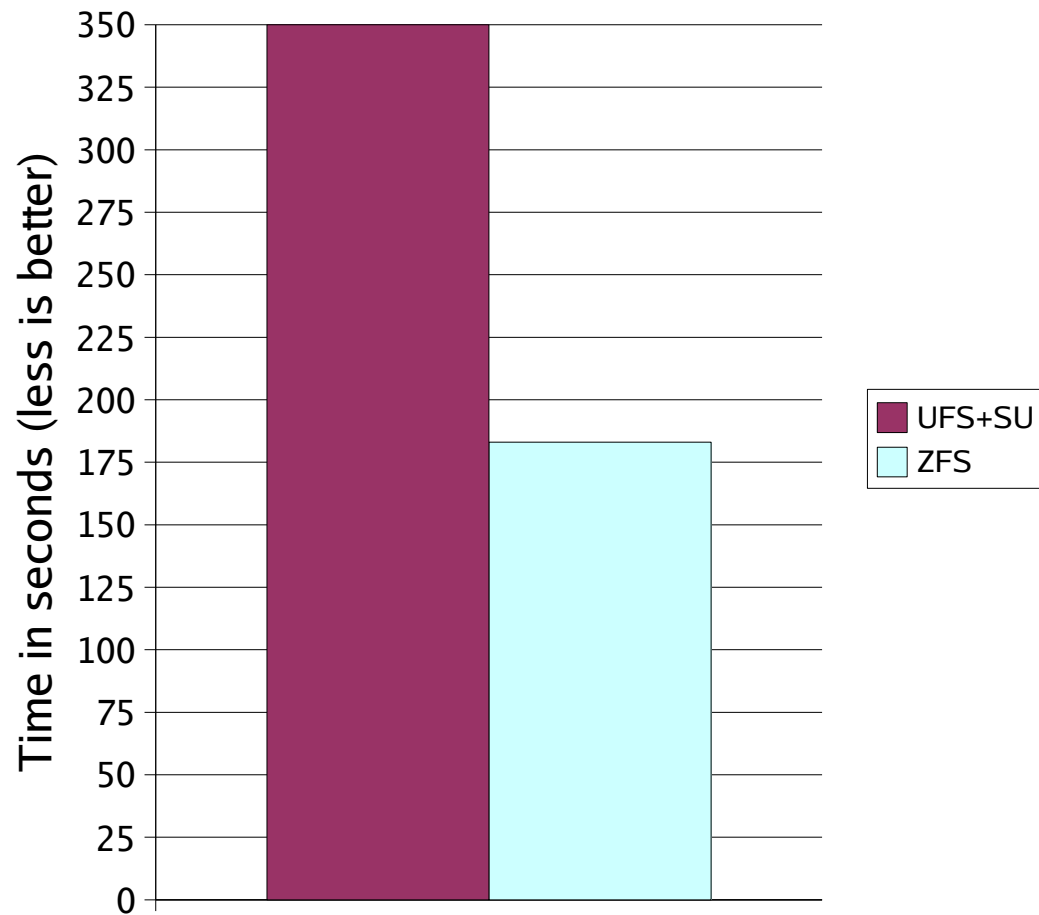
Untaring src.tar four times one by one

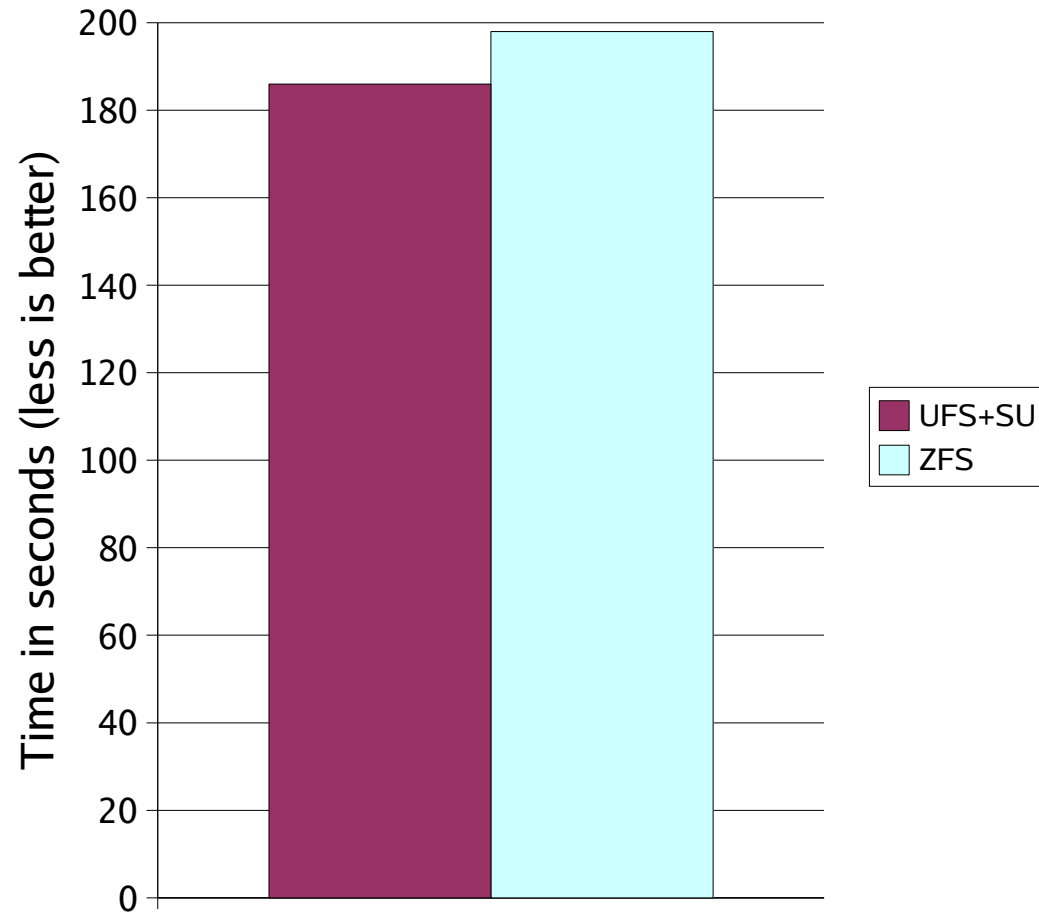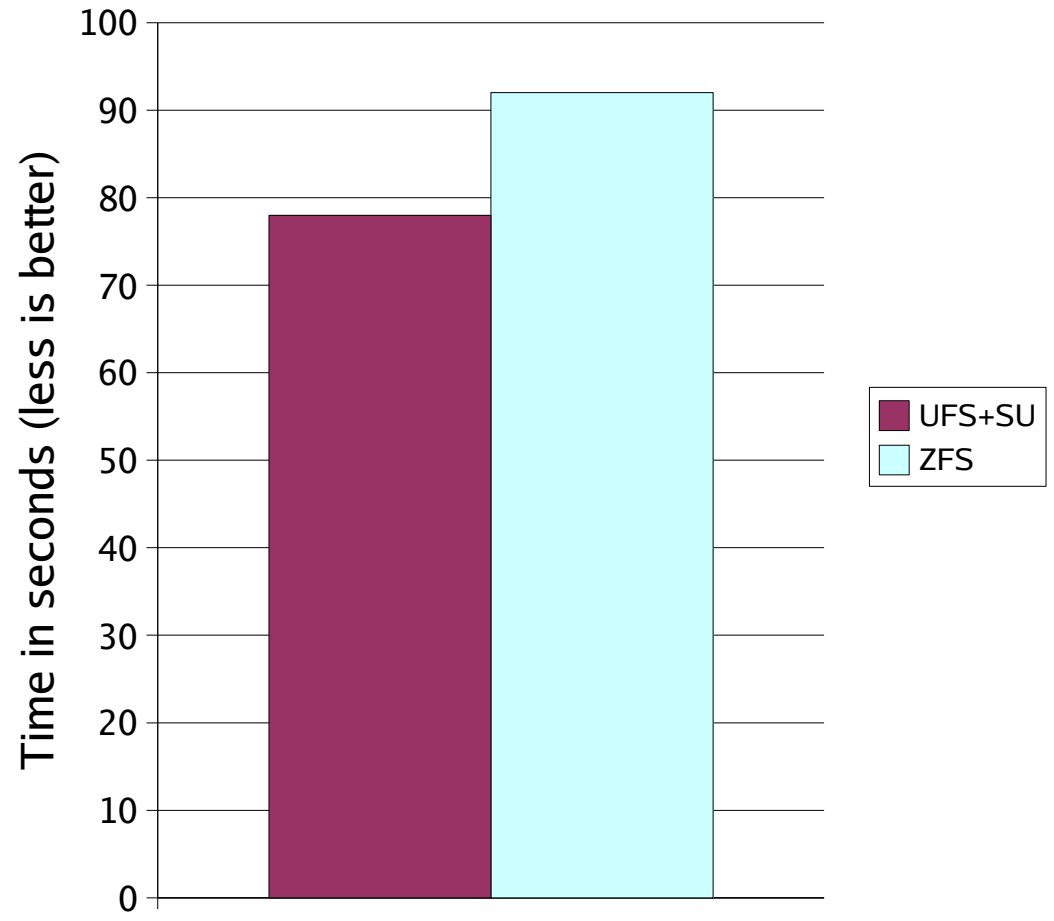# Removing four src directories one by one
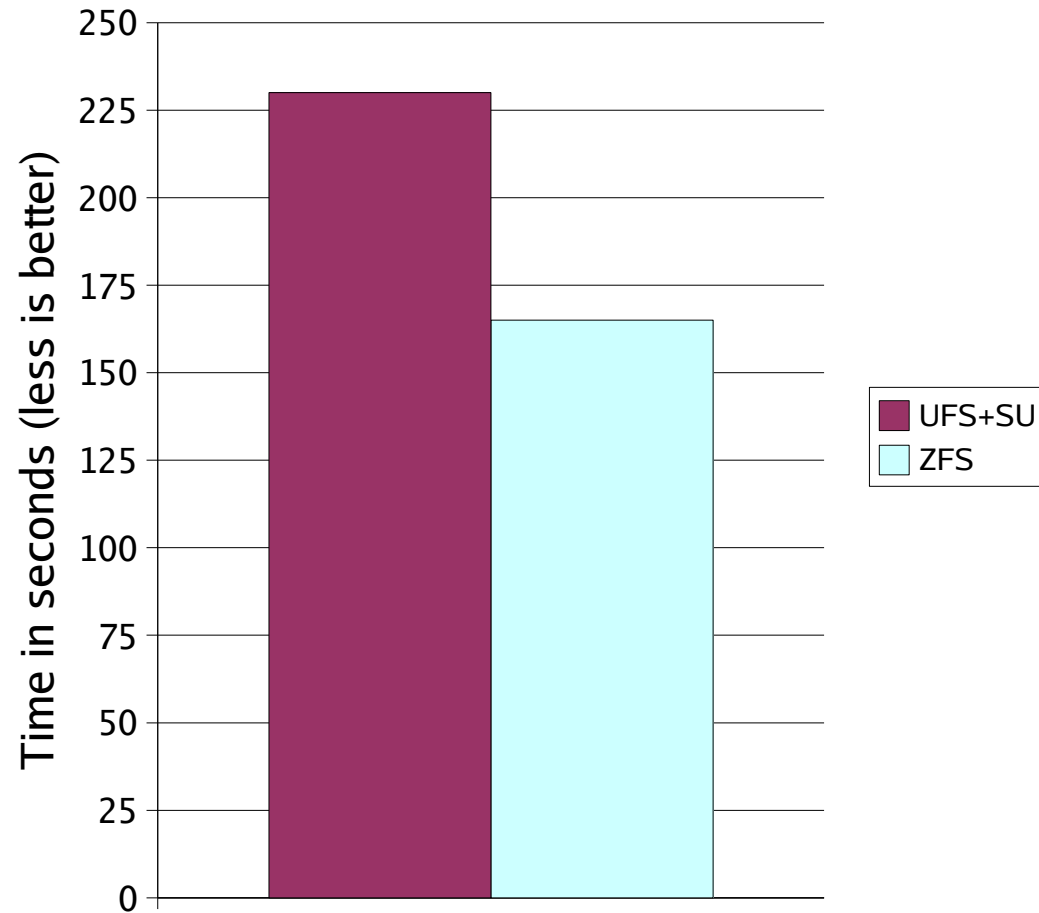
Untaring src.tar four times in parallel

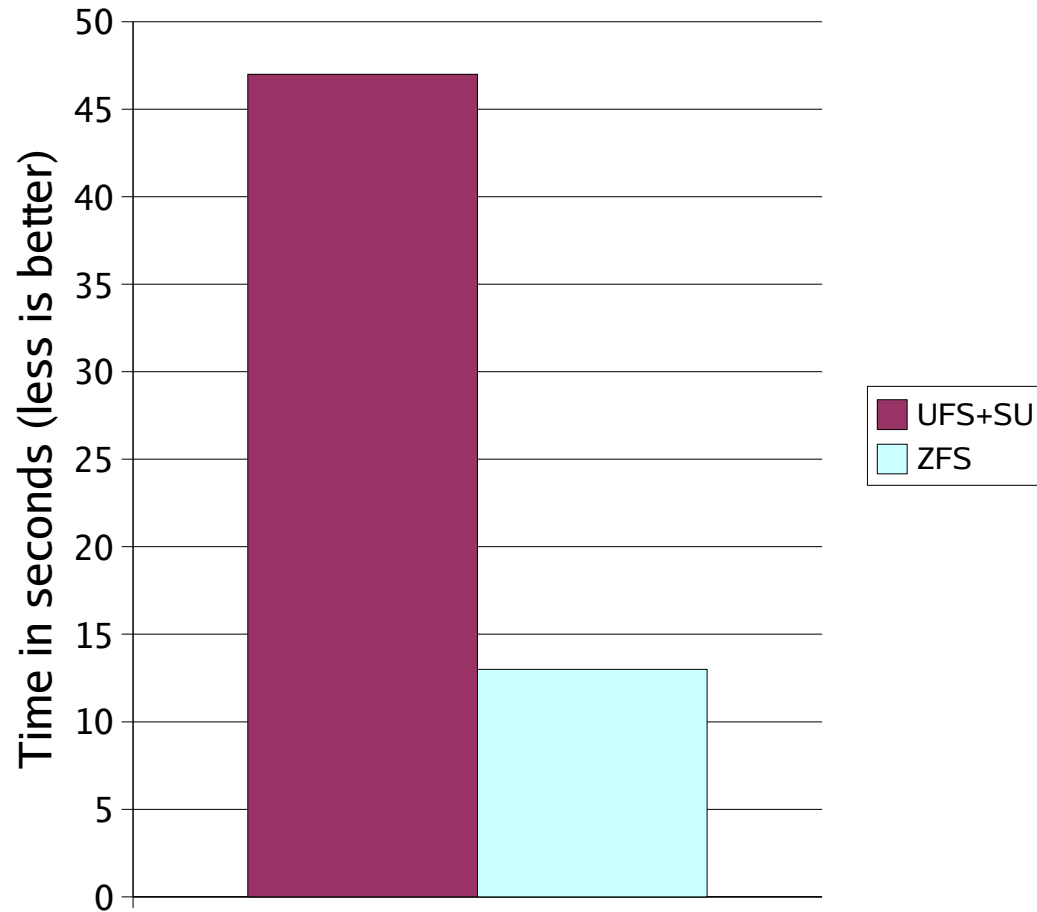# Removing four src directories in parallel

5GB of sequential write

4 x 2GB of sequential writes in parallel

fsx -N 50000 (operating on mmap(2)ed files)

Changes after initial commit

- rc.d/zfs startup script (by des@)
- periodic zfs script (by des@)
- support for amd64
- jails support
- reports via devd(8)
- root on ZFS
- hostid
- disk identifiers
- use of FreeBSD's namecache

Changes after initial commit

- performance improvements;
  based on help/work from ups@,
  jhb@, kris@, Attilio Rao
- many bug fixes; based on feedback
  from FreeBSD community

Changes in the pipeline

- extended attributes based on Solaris' fsattr(5)s
- support for other architectures

Future changes

- POSIX.1e ACLs based on extended attributes
- NFSv4-style ACLs
- iSCSI support for ZVOLs
- ZFS configuration at installation time
- booting from ZFS?

Some examples...