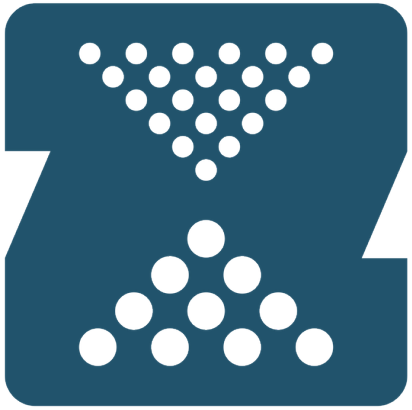


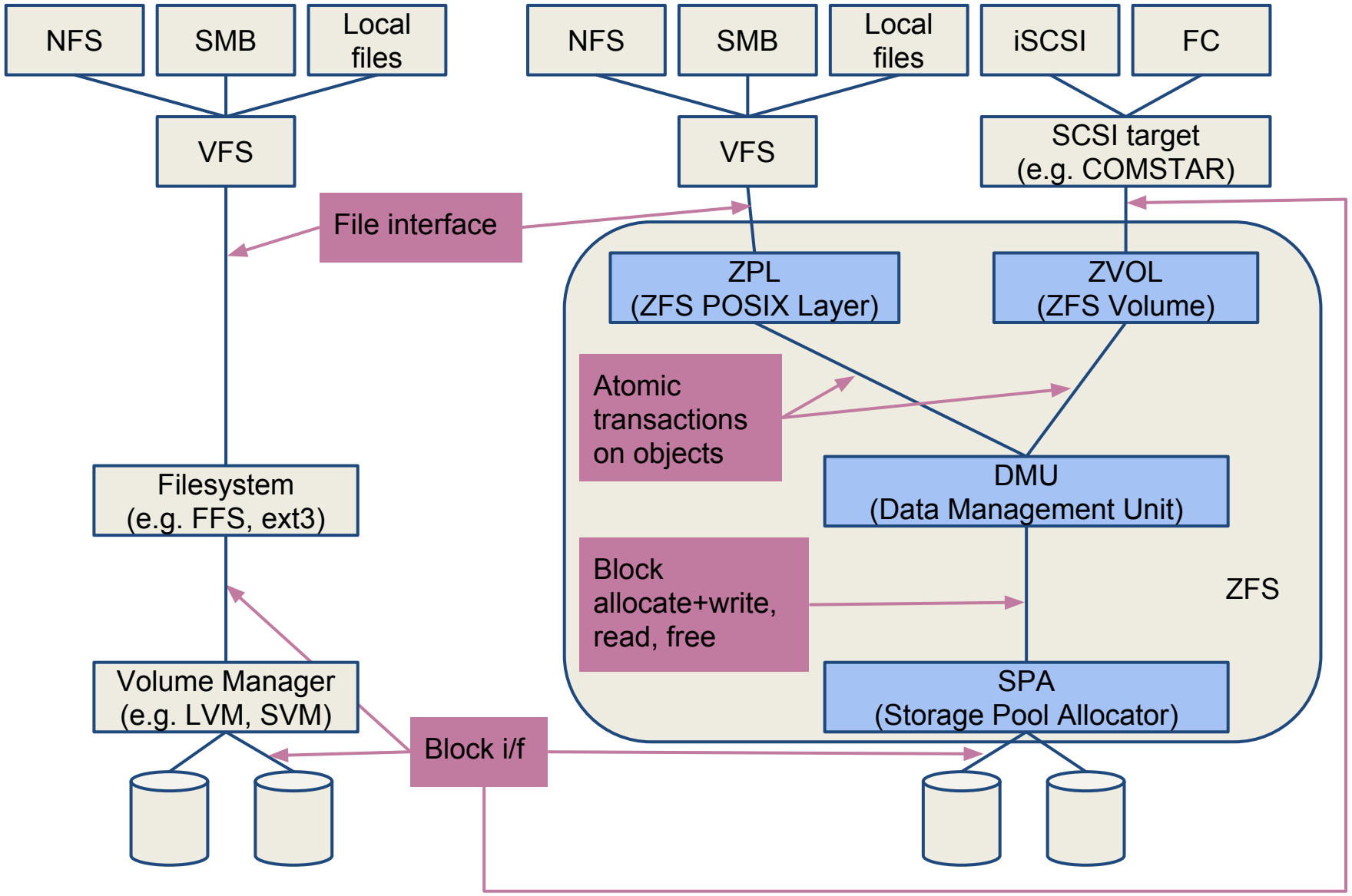
OpenZFS

Matt Ahrens
mahrens@delphix.com

- Most data (TB) in a pool?
- Most data in an all-flash pool?
- Most number of filesystems/zvols in a pool?
- Most number of snapshots in a pool?
- Most number of pools on one system?
- Largest ARC (GB)?
- Largest L2ARC cache (TB)?

What is the ZFS storage system?

- Pooled storage
  - Functionality of filesystem + volume manager in one
  - Filesystems allocate and free space from pool
- Transactional object model
  - Always consistent on disk (no FSCK, ever)
  - Universal - file, block, NFS, SMB, iSCSI, FC, …
- End-to-end data integrity
  - Detect & correct silent data corruption
- Simple administration
  - Concisely express intent
  - Scalable data structures


NFS
SMB
Local files
VFS
File interface
Filesystem (e.g. FFS, ext3)
Volume Manager (e.g. LVM, SVM)
Block i/f
NFS
SMB
Local files
iSCSI
FC
VFS
SCSI target (e.g. COMSTAR)
ZPL (ZFS POSIX Layer)
ZVOL (ZFS Volume)
Atomic transactions on objects
DMU (Data Management Unit)
Block allocate+write, read, free
SPA (Storage Pool Allocator)
ZFS

ZFS History

- 2001: development starts with 2 engineers
- 2005: ZFS source code released
- 2008: ZFS released in FreeBSD 7.0
- 2010: Oracle stops contributing to source code for ZFS
- 2010: illumos is founded as the truly open successor to OpenSolaris
- 2013: ZFS on (native) Linux GA
- 2013: Open-source ZFS bands together to form OpenZFS
- 2014: OpenZFS for Mac OS X launch

What is OpenZFS?

OpenZFS is a community project founded by open source ZFS developers from multiple operating systems:

- illumos, FreeBSD, Linux, OS X

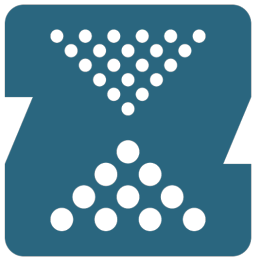
The goals of the OpenZFS project are:

- to **raise awareness** of the quality, utility, and availability of open source implementations of ZFS
- to encourage **open communication** about ongoing efforts to improve open source ZFS
- to ensure **consistent** reliability, functionality, and performance of all distributions of ZFS.

OpenZFS activities

http://open-zfs.org

- Platform-independent mailing list
  - Developers discuss and review platform-independent code and architecture changes
  - Not a replacement for platform-specific mailing lists
- Simplifying the illumos development process
- Creating cross-platform test suites
- Reducing code differences between platforms
- Office Hours a.k.a Ask the Expert



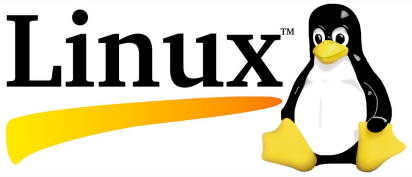
OpenZFS

Platform Diversity

stats on past 12 months (Sept 2012 - Aug 2013)

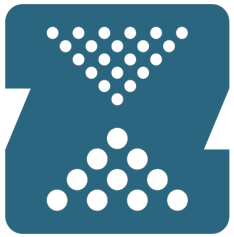
87 Commits
24 Contributors

229 Commits
19 Contributors



298 Commits
52 Contributors

379 Commits
5 Contributors



OpenZFS

illumos

Linux

Cloudius
the sky is not enough

FreeBSD

DELPHIX

Joyent

OmniTI

RACKTOP

nexenta
Enterprise class storage for everyone

Lawrence Livermore National Laboratory

debian

storiant

HELiOS

pogo [linux]

Scientific Linux

systems

SPECTRA

WHEEL SYSTEMS

HybridCluster

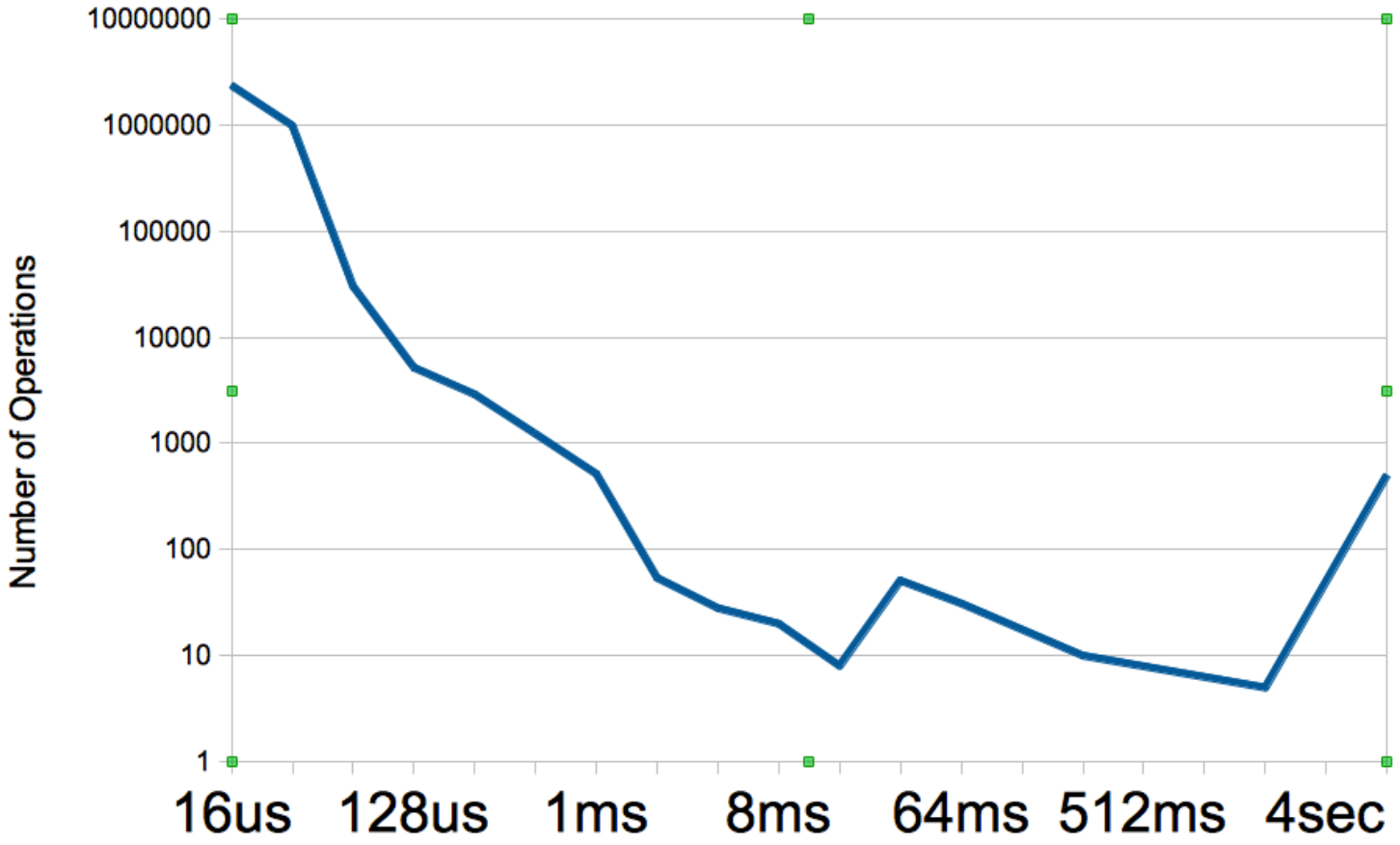
PCBSD

New in OpenZFS: Feature Flags

- How to version the on-disk format?
- Initial ZFS development model: all changes go through Sun
  - Linear version number
  - If support version X, must support all <X
- Feature flags enables independent development of on-disk features
- Independently-developed features can be later integrated into a common sourcebase

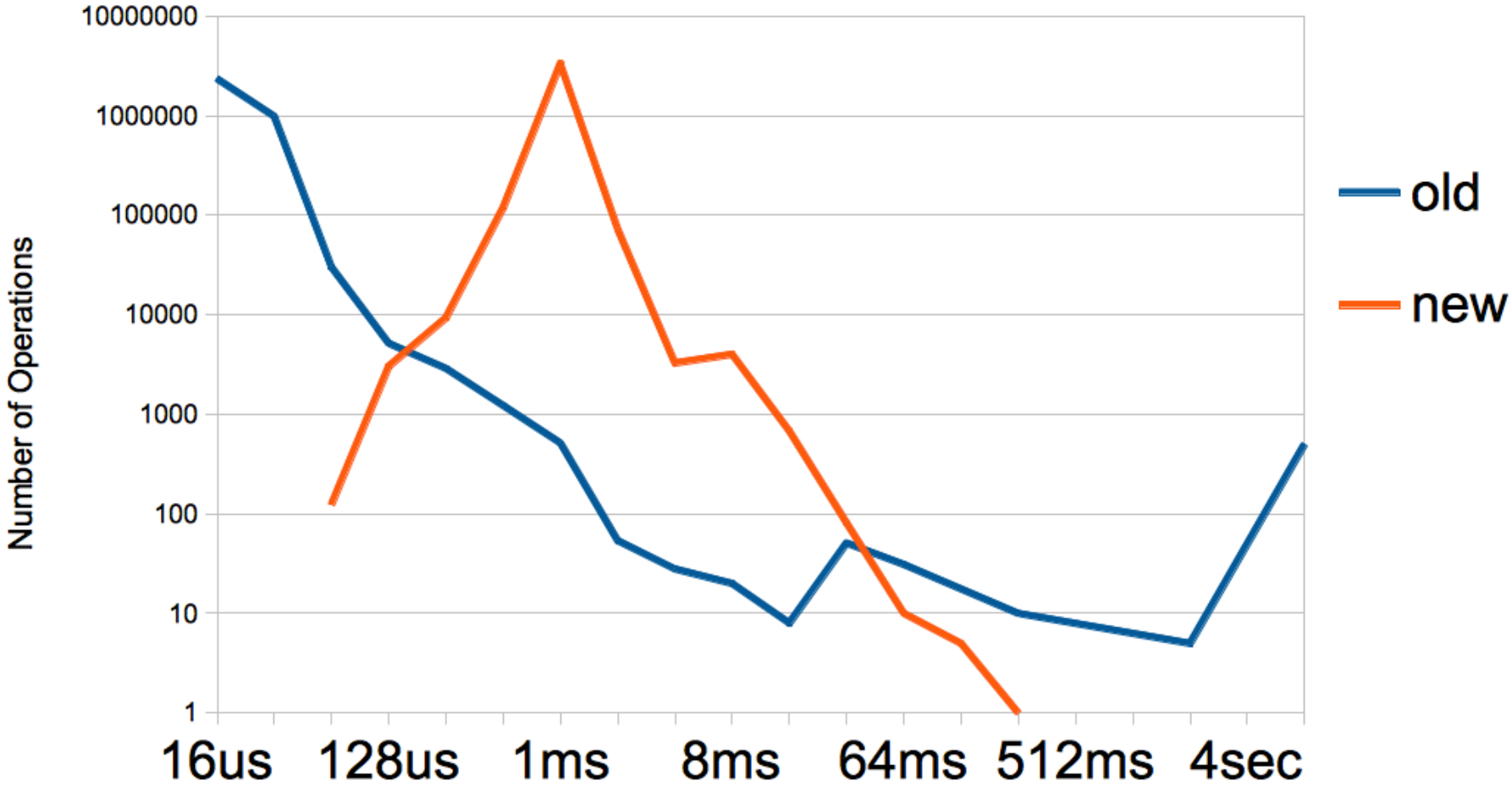
New in OpenZFS: Smoother Write Latency

- If application wants to write more quickly than the storage hardware can, ZFS must delay the writes
- old: 5,600 io/s; outliers: 10 seconds



New in OpenZFS: Smoother Write Latency

- old: 5,600 io/s; outliers: 10 seconds
- new: 5,900 io/s; outliers: 30 milliseconds

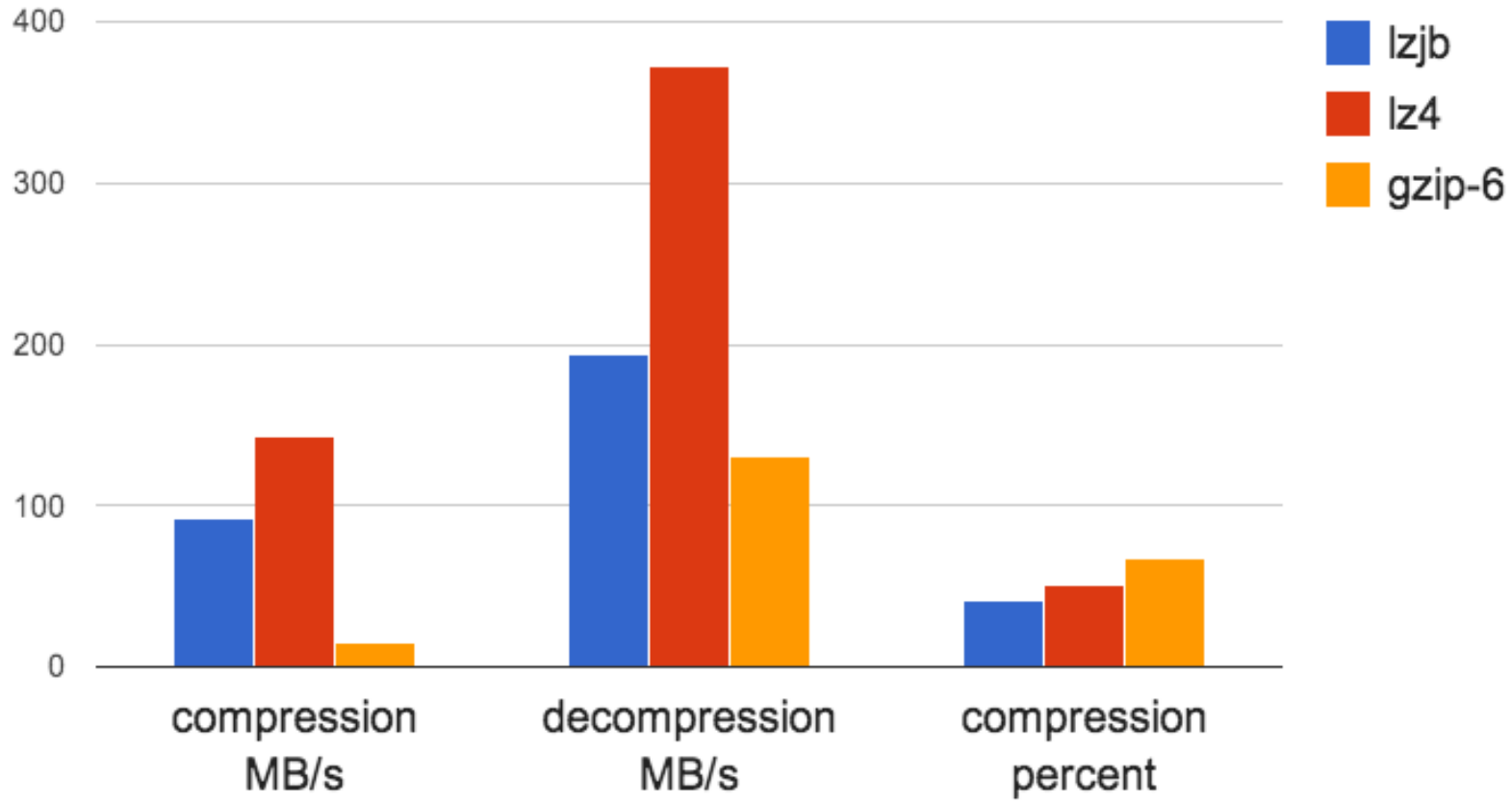


New in OpenZFS: Smoother Write Latency

- More info from Adam Leventhal's blog:
  - http://blog.delphix.com/ahl/2013/ zfs-fundamentals-write-throttle
  - http://blog.delphix.com/ahl/2014/ openzfs-write-throttle

New in OpenZFS: LZ4 compression

- Improved performance and compression ratio compared to previous default (lzjb)



Work in progress: Resumable send/receive

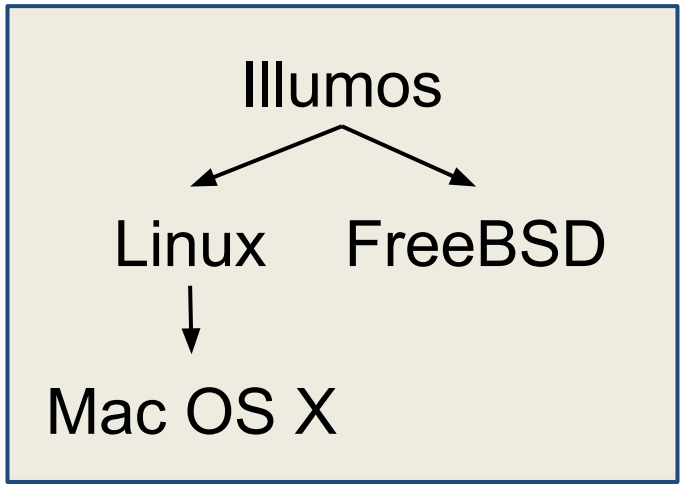
- send | receive is used for remote replication
- OpenZFS has zfs send progress reporting
- If system reboots, must restart from the beginning
- Solution: receiver remembers what data has been received, sender can restart from there

The future of OpenZFS: development model

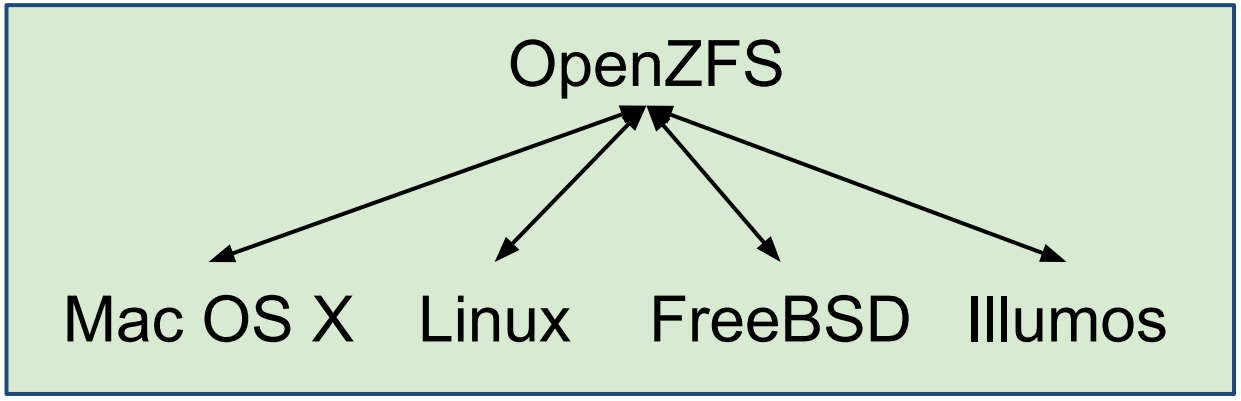
- Simplify getting changes into every platform
- Platform-independent codebase
  - all platforms pull from this verbatim, goal: no diffs
  - platform-independent changes pushed here first
- FreeBSD's and Linux's SPL will get less gross
- Illumos will get a (also non-gross) porting layer
- Only code that can be tested on any platform in userland
  - Test with ztest and TestRunner (formerly STF) tests
  - Will not include ZPL (posix layer) or vdev_disk

The future of OpenZFS: development model

Current



End Goal

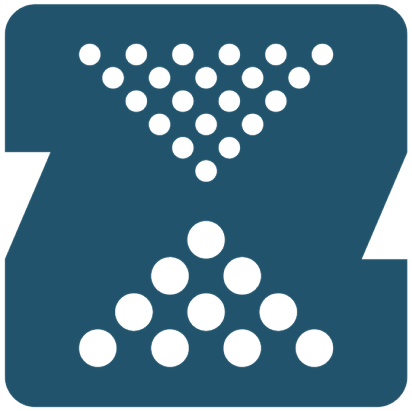


The future of OpenZFS: features

- platform-independent code repository
- hole birth time + other zfs send performance (integrated)
- filesystem & snapshot count limits (integrated)
- embedded block data (implemented)
- larger (1MB+) blocks support (implemented)
- more allocator performance (implemented)
- prefetch, ZIL, locking, UNMAP performance (Delphix)
- compressed ARC (George Wilson)
- persistent l2arc (Saso Kiselkov)
- performance on fragmented pools (George Wilson)
- resumable zfs send/recv (Delphix)
- device removal (Delphix)
- channel program for richer administration

How to get involved

- If you are making a product with OpenZFS
  - let us know, put logo on website & T-shirts
- If you are an OpenZFS admin/user
  - spread the word
- If you are writing code
  - join developer@open-zfs.org mailing list
  - attend 2nd annual OpenZFS Developer Summit
    - November 10-11, 2014, San Francisco
    - talk proposals due September 8th
  - get design help or feedback on code changes
  - take a look at project ideas!



OpenZFS

http://open-zfs.org

Matt Ahrens
mahrens@delphix.com

Features unique to OpenZFS

- Feature Flags
- libzfs_core
- CLI Usability
  - size estimates for zfs send and zfs destroy
  - vdev information in zpool list
  - zfs send progress reporting
  - arbitrary snapshot arguments to zfs snapshot
- Dataset properties
  - refcompressratio
  - clones
  - written, written@*snap*
  - lused, lcompressed
- TestRunner test suite

Performance improvements in OpenZFS

- async filesystem and volume destruction
- single-copy ARC cache
- space allocation (spacemap) performance improvements
- smoother write latency (write throttle rewrite)
- per-type i/o queues (read, ZIL, async write, scrub)
- lz4 compression
- compressed cache devices (L2ARC)

OpenZFS development process - illumos

How to develop changes:

- Large changes: review on developer@open-zfs.org
- Set up OpenIndiana-based dev environment
- Clone repo from github.com/illumos
- Make code changes
- Run a full build with nightly (runs lint)
- Test with ztest and TestRunner
  - consider if you need to add a new test case
- Check code style with cstyle tool

OpenZFS development process - illumos (2)

How to submit code:

- Get your code reviewed on zfs@lists.illumos.org
  - cc: developer@open-zfs.org for platform-neutral changes
  - non-trivial changes typically must be reviewed by a ZFS expert (e.g. Matt Ahrens or George Wilson)
  - preferred tool for creating reviews is webrev
- Submit a "Request to Integrate" (RTI) email to advocates@lists.illumos.org
- Advocate will integrate (push) your code to github
  - Chris Siden is the most active ZFS advocate

OpenZFS development process - FreeBSD

Code base: FreeBSD SVN tree

Porting process:

1. pulling code changes from illumos to vendor branch
   - vendor/illumos/dist
2. MFV to head
   - kernel: head/sys/cddl/contrib/opensolaris
   - userland: head/cddl/contrib/opensolaris
3. MFC to stable after a grace period

Solaris porting layer: head(/sys)/cddl/compat/opensolaris

OpenZFS development process - FreeBSD (2)

Basic rules:

- keep vendor's directory structure
- keep as close to vendor (illumos) as possible
- mark changed or different code

Challenges:

- backward (and forward) compatibility
- FreeBSD-specific differences
  - boot loader, GEOM integration, FreeBSD jails, VM / VFS integration

OpenZFS development process - FreeBSD (3)

How to submit code:

- Platform-independent changes -> illumos
- FreeBSD-specific changes -> head branch
- Exemption: platform-independent critical bugfixes go direct to head + should be reported to illumos
- Discuss changes on the freebsd-fs@ mailing list
  - zfs-devel@ for developers

OpenZFS development process - Linux

Code Base: github

- independent code base (not in mainline kernel)
- divided into spl (solaris porting layer) and zfs
- atm. mainly linux-specific activity
- behind recent illumos code base

Submitting changes:

- github pull requests / issue tracker

Work in progress: Large block support

- Good ideas come from all sorts of places
- Proprietary (Oracle) ZFS has 1MB block support
- Improves performance, especially for RAID-Z w/4k devices
- Ideally, OpenZFS will provide compatibility with proprietary on-disk format