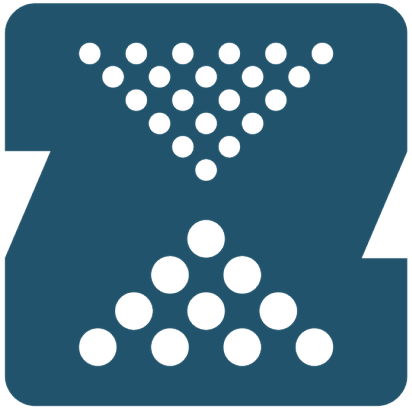


OpenZFS

Matt Ahrens
mahrens@delphix.com

Brian Behlendorf
behlendorf1@llnl.gov

DELPHIX

Lawrence Livermore National Laboratory

LLNL-PRES-643675
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

LinuxCon 2013

September 17, 2013

Brian Behlendorf, Open ZFS on Linux

Lawrence Livermore National Laboratory

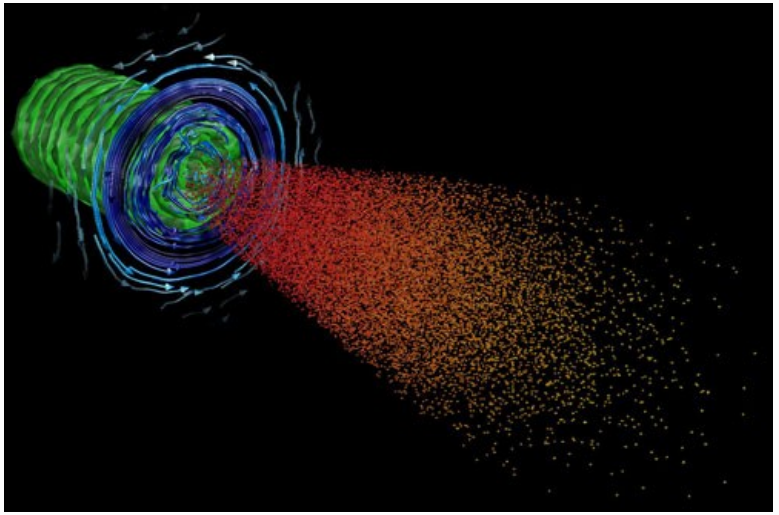
LLNL-PRES-643675

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract

DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

High Performance Computing

- Advanced Simulation
  - Massive Scale
  - Data Intensive
- Top 500
  - #3 Sequoia
    - 20.1 Peak PFLOP/s
    - 1,572,864 cores
    - 55 PB of storage at 850 GB/s
  - #8 Vulcan
    - 5.0 Peak PFLOPS/s
    - 393,216 cores
    - 6.7 PB of storage at 106 GB/s



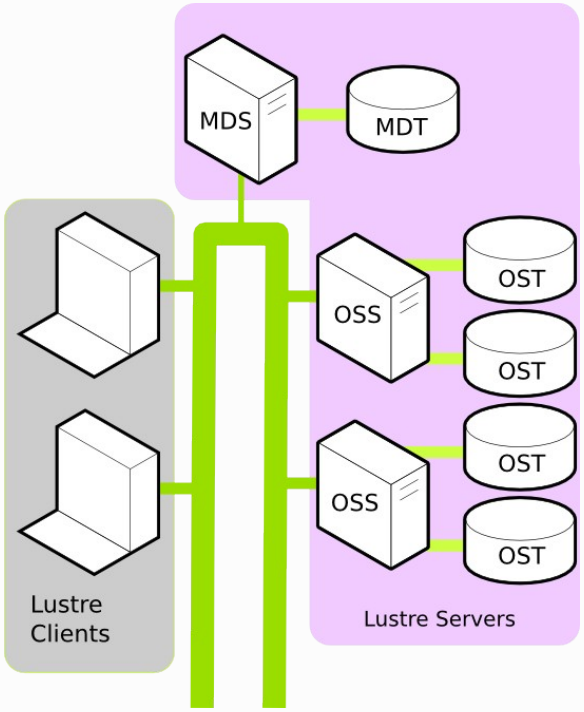
World class computing resources

Linux Clusters

- First Linux cluster deployed in 2001
- Near-commodity hardware
- Open Source
- Clustered High Availability Operating System (CHAOS)
  - Modified RHEL Kernel
  - New packages – monitoring, power/console, compilers, etc
  - Lustre Parallel Filesystem
  - ZFS Filesystem

Lustre Filesystem

- Massively parallel distributed filesystem
- Lustre servers use a modified ext4
  - Stable and fast, but...
  - No scalability
  - No data integrity
  - No online manageability
- Something better was needed
  - Use XFS, BTRFS, etc?
  - Write a filesystem from scratch?
  - Port ZFS to Linux?



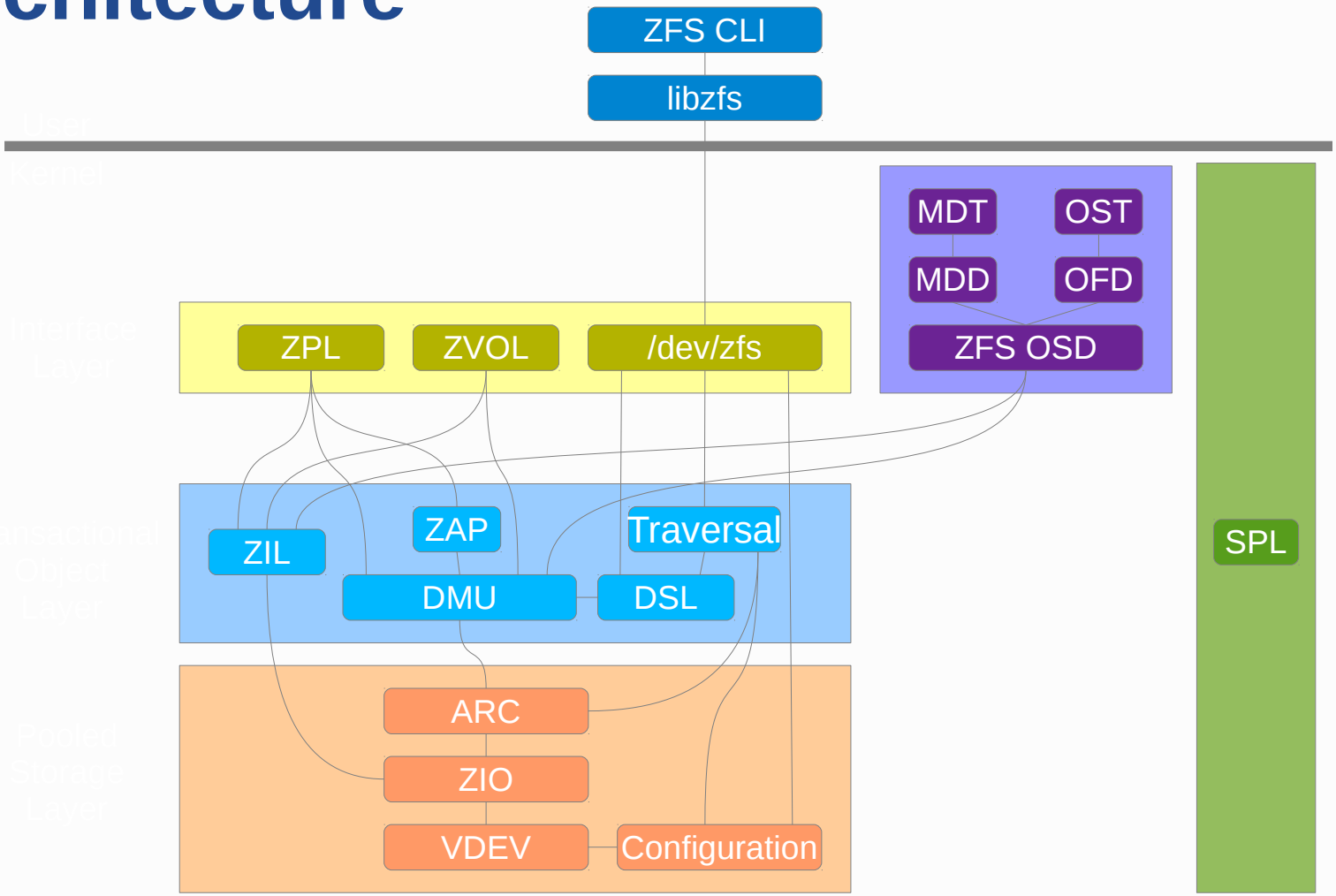
Existing Linux filesystems do not meet our requirements

ZFS on Linux History

- 2008 – Prototype to determine viability
- 2009 – Initial ZVOL and Lustre support
- 2010 – Development moved to Github
- 2011 – POSIX layer added
- 2011 – Community of early adopters
- 2012 – Production usage of ZFS
- 2013 – Stable GA release

Community involvement was exceptionally helpful

Architecture



Linux Specific Changes

- Core ZFS code is self contained
  - May be built in user space or kernel space
  - Includes functionality for snapshots, clones, I/O pipeline, etc
- Solaris Porting Layer
  - Adds stable Solaris/Illumos interfaces
    - Taskqs, lists, condition variables, rwlocks, memory allocators, etc...
  - Layered on top of Linux equivalents if available
  - Solaris specific interfaces were implemented from scratch
- Vdev disk
  - Interfaces with the Linux kernel block layer
  - Had to be rewritten to use native Linux interfaces

The core ZFS code required little change

Linux Specific Changes

- Interface Layer
  - /dev/zfs
    - Device node interface for user space zfs and zpool utilities
    - Minor changes needed for a Linux character device
  - ZVOL: ZFS Volumes
    - Reimplemented as Linux block driver which is backed by the DMU
  - ZPL: ZFS Posix Layer
    - Most complicated part, there are significant VFS differences
    - In general new functions were added for the Linux VFS handlers
    - If possible the Linux handlers use the equivalent Illumos handler
  - Lustre
    - Support for Lustre was added by Sun/Oracle/Whamcloud/Intel
    - Lustre directly layers on the DMU and does not use the Posix Layer

The majority of changes to ZFS were done in the interface layer

Porting Issues

- 8k stacks
  - Illumos allows larger stacks
  - Needed to get stack usage down to support stock distribution kernels
  - Code reworked as needed to save stack
- Gcc
  - ZFS was written for C99, Linux kernel is C89
  - Fix numerous compiler warnings
- GPL-only symbols
  - ZFS is under an open source license but may not use all exported symbols
  - This includes basic functionality such as work queues
  - ZVOLs can't add entries in /sys/block/
  - .zfs/snapshot's can't use the automounter
- User space
  - Solaris threads used instead of pthreads
  - Block device naming differences
  - Udev integration

Porting Issues

- Memory management
  - ZFS relies heavily on virtual memory for its data buffers
    - By design the Linux kernel discourages the use of virtual memory
    - To resolve this the SPL provides a virtual memory based slab
      - This allows use of the existing ZFS IO pipeline without modification
      - Fragmentation results in underutilized memory
      - Stability concerns under low memory conditions
    - We plan to modify ZFS to use scatter-gather lists of pages under Linux
    - Allows larger block sizes
    - Allows support for 32-bit systems
  - The ARC is not integrated with the Linux page cache
    - Memory used by the ARC is not reported as cached pages
    - Complicates reclaiming memory from the ARC when needed
    - Requires an extra copy of the data for mmap'ed I/O

Future Work

- Features
  - O_DIRECT
  - Asynchronous IO
  - POSIX ACLs
  - Reflink
  - Filefrag
  - Fallocate
  - TRIM
  - FMA Infrastructure (event daemon)
  - Multiple Modified Protection (MMP)
  - Large blocks

Possibilities for future work

Source Code

- All source code and the issue tracker are kept at Github
  - http://github.com/zfsonlinux/zfs
  - 70 contributors, 171 forks, 816 watchers
- Not in mainline kernel
  - Similar to resier4, unionfs, lustre, ceph, …
  - Autotools used for compatibility
  - Advantages
    - One code base used for 2.6.26 - 3.11 kernels
    - Users can update ZFS independently from the kernel
    - Simplifies keeping in sync with Illumos and FreeBSD
    - User space utilities and kernel modules can share code
  - Disadvantages
    - Support for the latest kernel lags slightly
    - Higher maintenance and testing burden for developers

Where is ZFS on Linux Today

- Stable and used in production
- Performance is comparable to existing Linux filesystems
- All major ZFS features are available
  - Simplified administration - Stripes, Mirrors, and RAIDZ[1,2,3]
  - Online management - ZFS Intent Log (ZIL)
  - Snapshots / Clones - L2ARC Tiered Caching
  - Special .zfs directory - Transparent compression
  - Send/receive of snapshots - Transparent deduplication
  - Virtual block devices (ZVOL)
- Currently used by Supercomputers, Desktops, NAS appliances
- Enthusiastic user community
  - zfs-discuss@zfsonlinux.org

ZFS is available on Linux today!

ZFS History

- 2001: development starts with 2 engineers
- 2005: ZFS source code released
- 2006: ZFS on FUSE for Linux started
- 2008: ZFS released in FreeBSD 7.0
- 2008: ZFS on (native) Linux port started
- 2008: Sun's 7000 series ZFS Storage Appliance ships
- 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

What is OpenZFS?

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

- illumos, FreeBSD, Linux, OS X

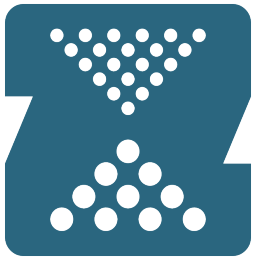
The goals of the OpenZFS project are:

- 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



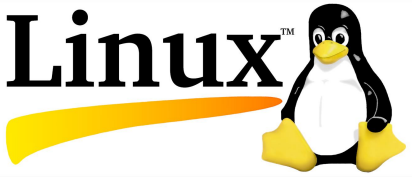
OpenZFS

Platform Diversity

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

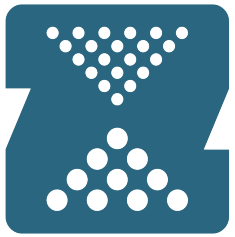
87 Commits
24 Contributors

229 Commits
19 Contributors

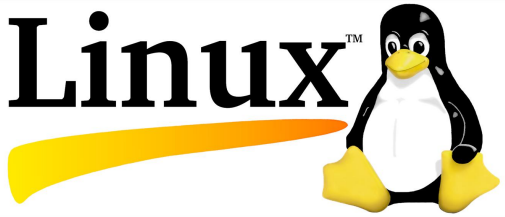


298 Commits
52 Contributors

379 Commits
5 Contributors



OpenZFS



DELPHIX

Joyent

OmniTI

RACKTOP

nexenta
Enterprise class storage for everyone

Lawrence Livermore National Laboratory

debian

gentoo

HELiOS

sagecloud

pogo [linux]™

Scientific Linux

systems

SPECTRA

WHEEL SYSTEMS

HybridCluster

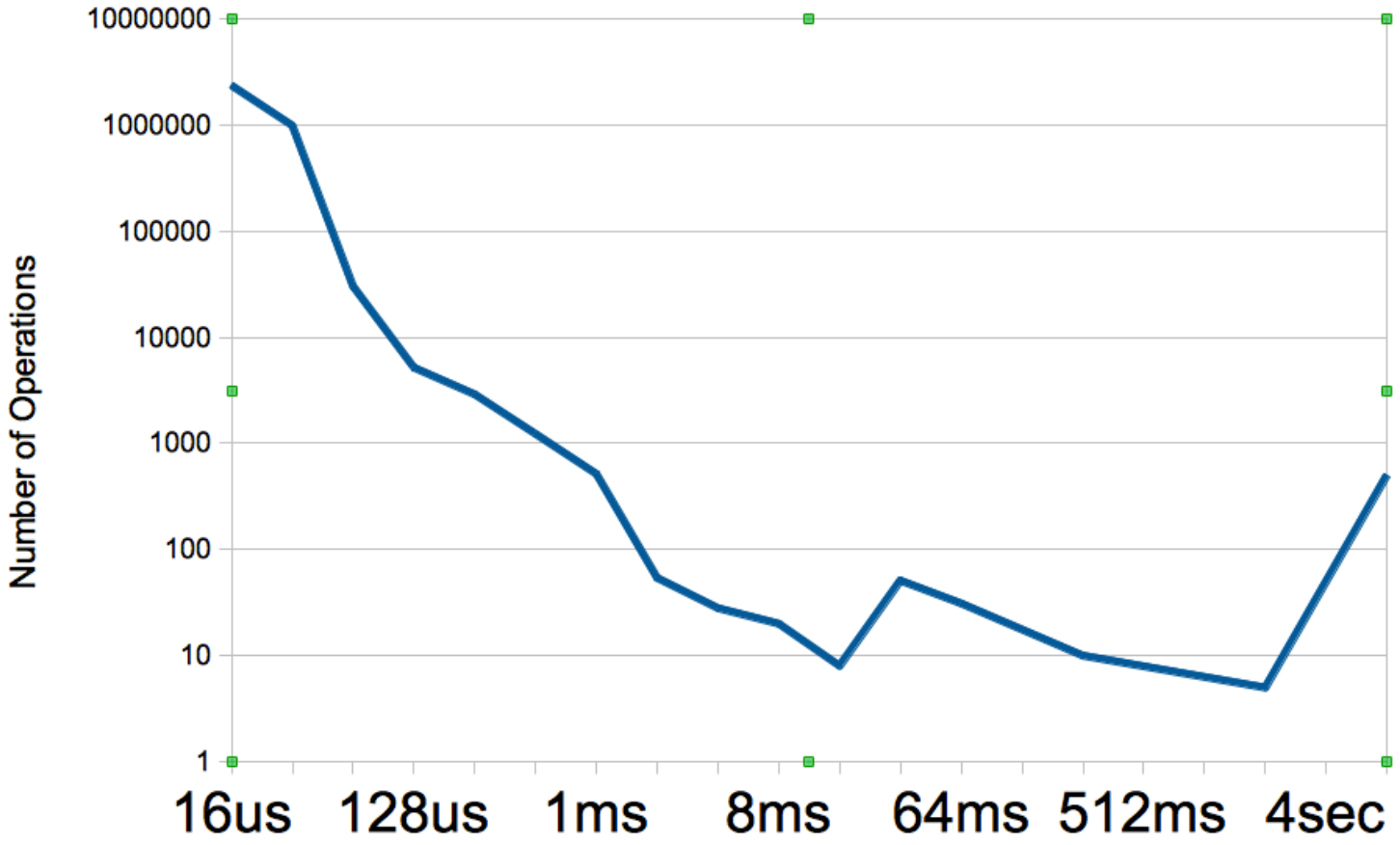
PCBSD

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 versions <X
- Feature flags enables independent development of on-disk features
- Independently-developed features can be later integrated into a common sourcebase

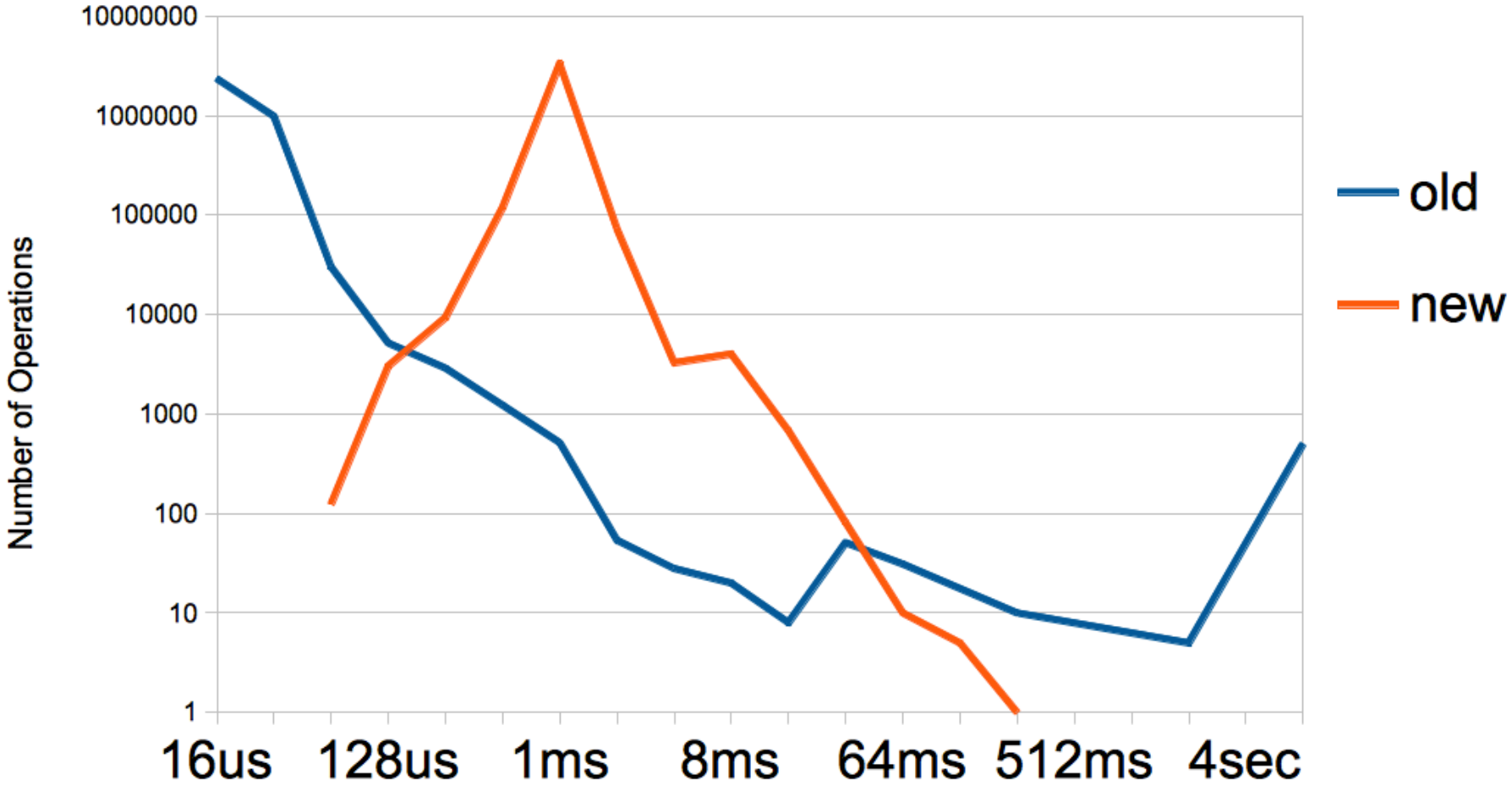
New in OpenZFS: Smoother Write Latency

- 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



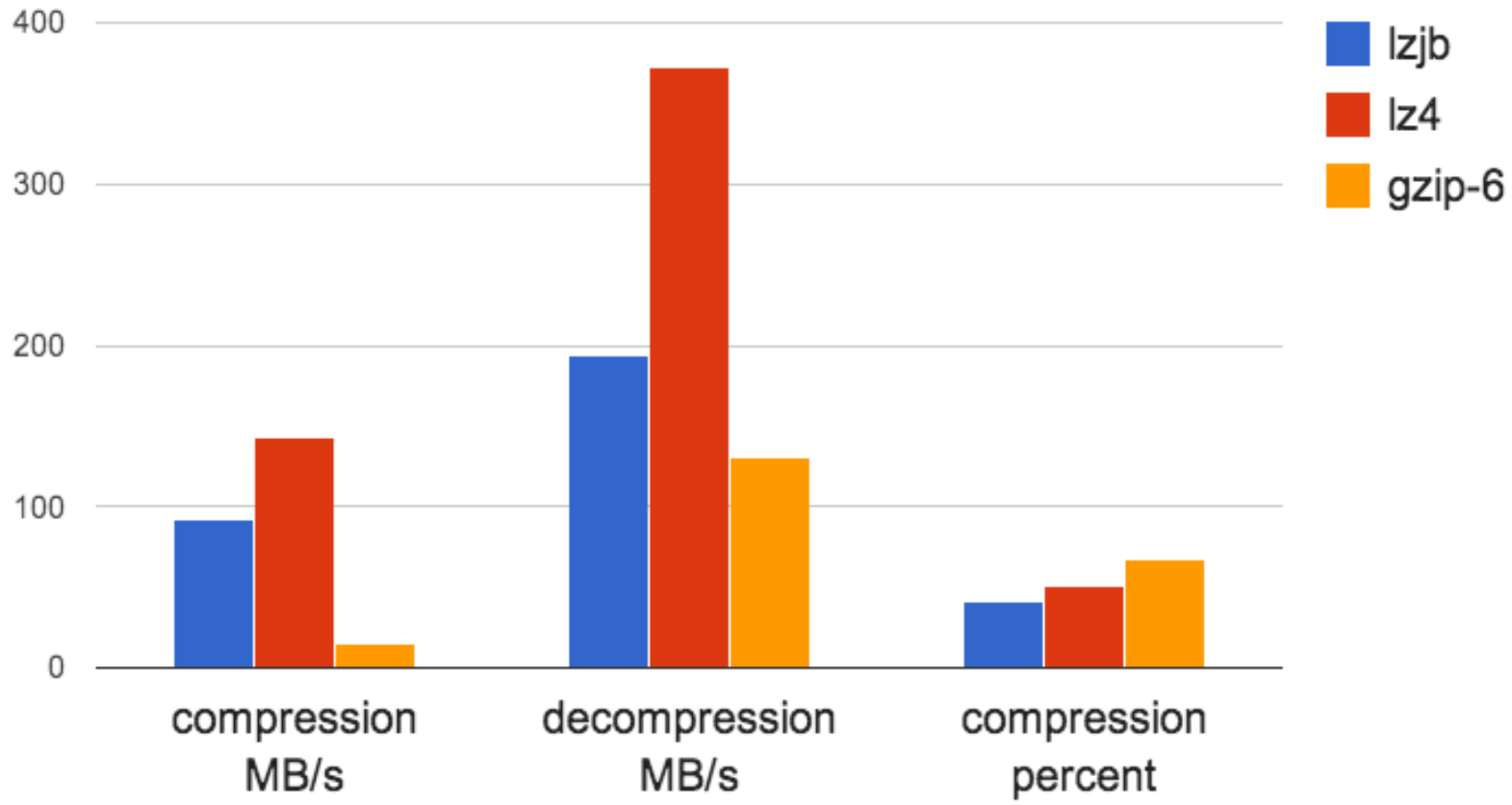
New in OpenZFS: Smoother Write Latency

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



New in OpenZFS: LZ4 compression

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



The future of OpenZFS: collaboration

- Office Hours a.k.a Ask the Expert
- Reduce code differences between platforms
  - most diffs will then apply cleanly to all platforms
- Cross-platform test suite
- More complete userland implementation
  - Allow running /sbin/zfs & /sbin/zpool against libzpool
  - Could enable platform-independent upstream repo
- Separate ZPL into platform-specific and platform-independent layers
- Create virtual machine images of each platform to enable easier cross-platform testing

Future of OpenZFS: 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 “bookmark”, sender can restart from bookmark

Future of OpenZFS: 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

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)

The future of OpenZFS: features

- persistent l2arc (Saso Kiselkov)
- performance on fragmented pools (George Wilson)
- observability -- zfs dtrace provider
- resumable zfs send/recv (Chris Siden)
- filesystem & snapshot count limits (Jerry Jelinek)
- device removal?
- revived MacOS port (Jorgen Lundman)
- Larger (1MB+) block support
- multi-modifier protection
- large dnodes (to fit more attributes w/o spill block)
- channel program for richer administration (Max Grossman)
- Raspberry pi support for ZFS on Linux (Richard Yao)

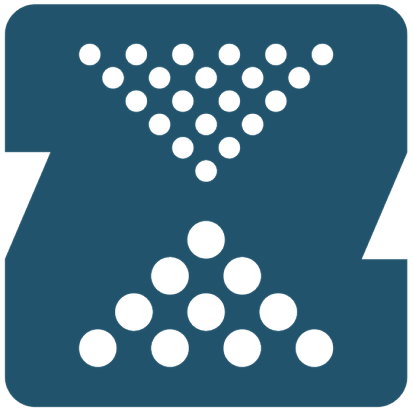
The future of OpenZFS: development model

- Platform-independent codebase
  - all platforms pull from this verbatim, goal: no diffs
  - platform-independent changes pushed here first
- Only code that can be tested on any platform in userland
- Some code is tested in userland today by ztest
  - but not libzfs, send/recv, properties, delegated administration (zfs allow), etc. etc.
- Need ioctl layer so that /sbin/zfs can run against userland impementation
- Need to get TestRunner tests (ported from STF) running against userland
- Some way to run the platform-independent parts of the ZPL?

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
  - contribute to documentation wiki on open-zfs.org
- If you are writing code
  - join developer@open-zfs.org mailing list
  - get design help or feedback on code changes
  - take a look at project ideas!

DELPHIX



OpenZFS

http://open-zfs.org

Matt Ahrens
mahrens@delphix.com

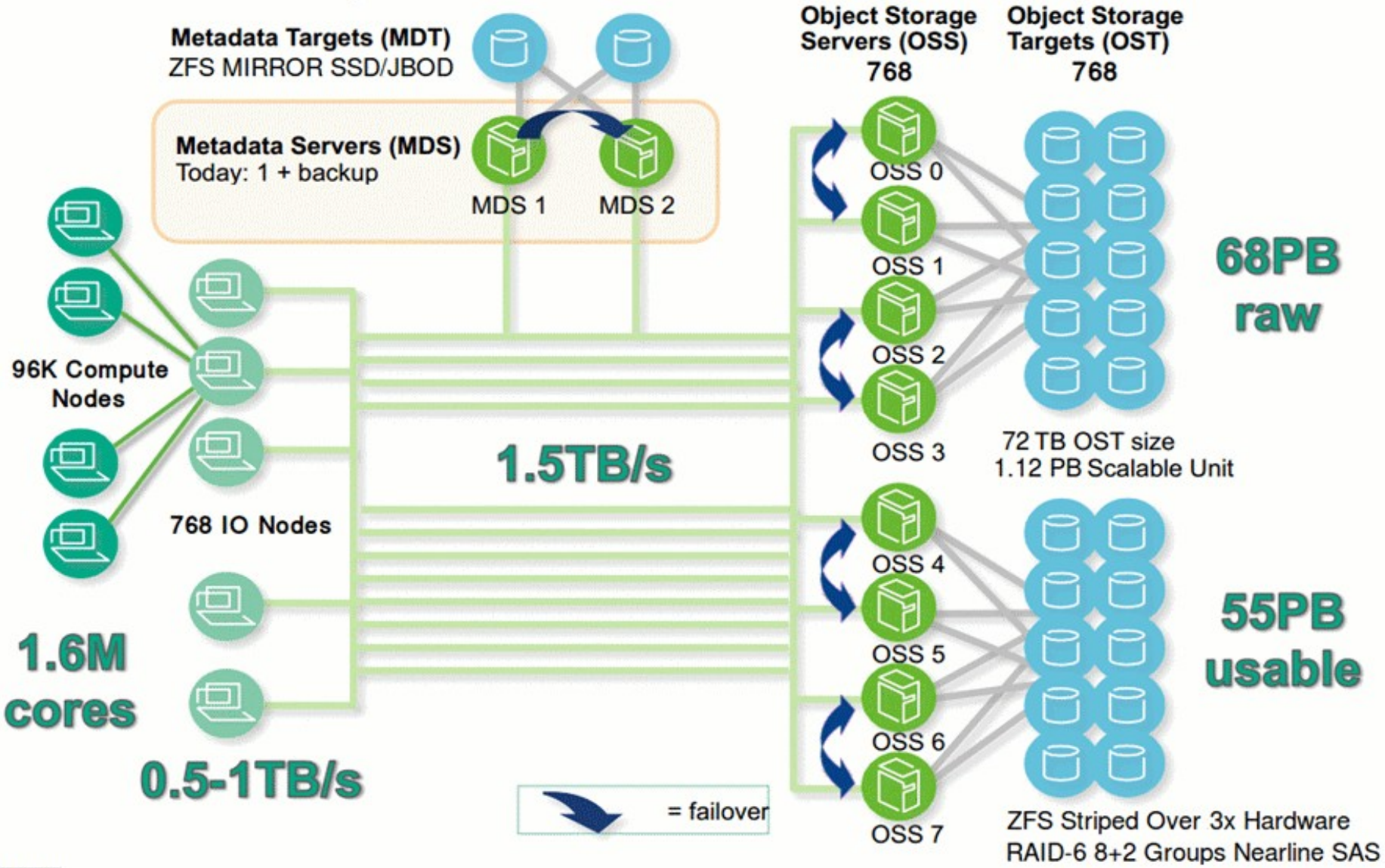
Brian Behlendorf
behlendorf1@llnl.gov

Licensing

- ZFS is Open Source under the CDDL
- ZFS is NOT a derived work of Linux
  - “It would be rather preposterous to call the Andrew FileSystem a 'derived work' of Linux, for example, so I think it's perfectly OK to have a AFS module, for example.”
    - Linus Torvalds
  - “Our view is that just using structure definitions, typedefs, enumeration constants, macros with simple bodies, etc., is NOT enough to make a derivative work. It would take a substantial amount of code (coming from inline functions or macros with substantial bodies) to do that.”
    - Richard Stallman

ZFS can be used in Linux

LLNL Sequoia Lustre Architecture



LLNL-PRES-582221