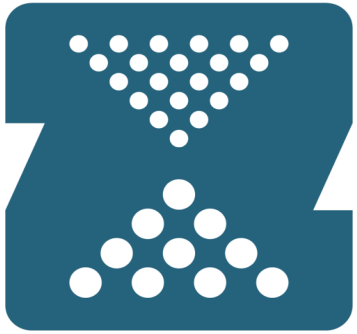


OpenZFS

Andrew Gabriel
Cucumber Technology Ltd
andrew@cucumber.me.uk

UKOUG
UK ORACLE USER GROUP

17th June 2015

What is ZFS?

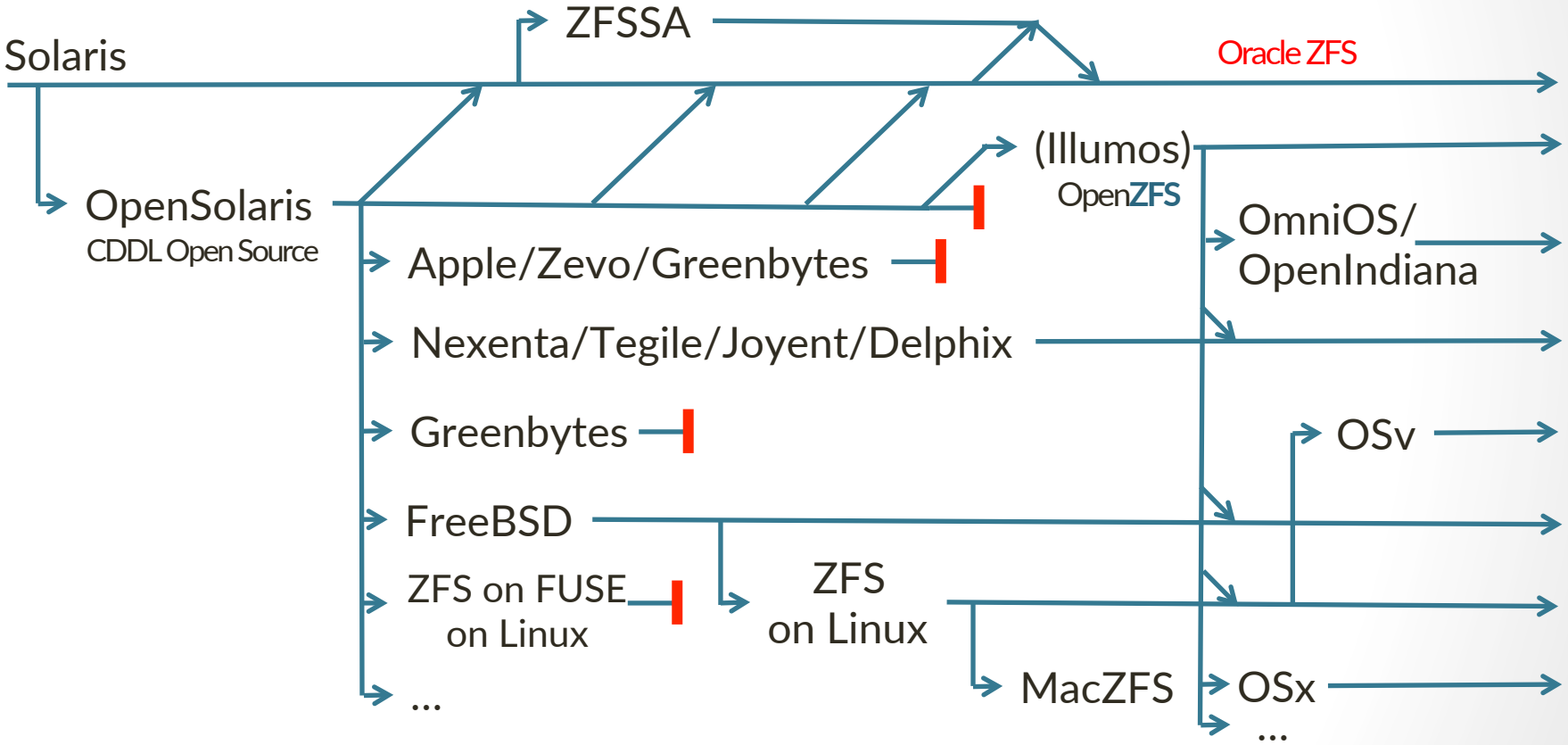
New file system developed by Sun Microsystems, starting development in 2001, open sourced 2005, released 2006.

- Built-in data integrity for exponentially increasing data volumes with fixed disk error rates.

Got Checksums?

- Transactional – always consistent on disk – no fsck(1M)
- No size limits.
- Simplify storage management. Everything shares a pool (like system memory). No separate volume management.
- Performance.

ZFS Ports History



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 Platform Diversity

DELPHIX
Joyent
OmniTI
openindiana
RACKTOP
nexenta
tegile
SYNETO

98 Commits
34 Contributors

HybridCluster
SPECTRA
iXsystems
FreeNAS
PCBSD
WHEEL SYSTEMS

298 Commits
52 Contributors

Lawrence Livermore National Laboratory
CentOS openSUSE debian fedora gentoo linux ubuntu
Scientific Linux HELiOS pogo linux
storiant
Fraunhofer FS BeeGFS developed by Fraunhofer Zetavault
OSNEXUS STORAGE MADE SIMPLE
cloudscaling (EMC)

372 Commits
33 Contributors

879 Commits
33 Contributors

OSv
designed for the cloud

Cloudius
the sky is not enough

Commits and Contributors (with at least 2 commits), figures are over a 1 year period

ZFS on Linux on Raspberry Pi

Here's ZFS built, and running on a Raspberry Pi !

The picture shows ZFS running on a Pi B+ (512MB memory).

I rebuilt it on the new Pi 2 (1GB memory, 4-core).

The JBOD consists of 7 USB thumb drives on a USB hub.

OpenZFS

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

OpenZFS – Feature Flags

- How to version the on-disk format with multiple developers?
- 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
- ZFS implementations know how to safely handle features they don't implement

com.delphix:async_destroy
com.delphix:empty_bpobj
org.illumos:lz4_compress
com.delphix:spacemap_histogram
com.joyent:multi_vdev_crash_dump

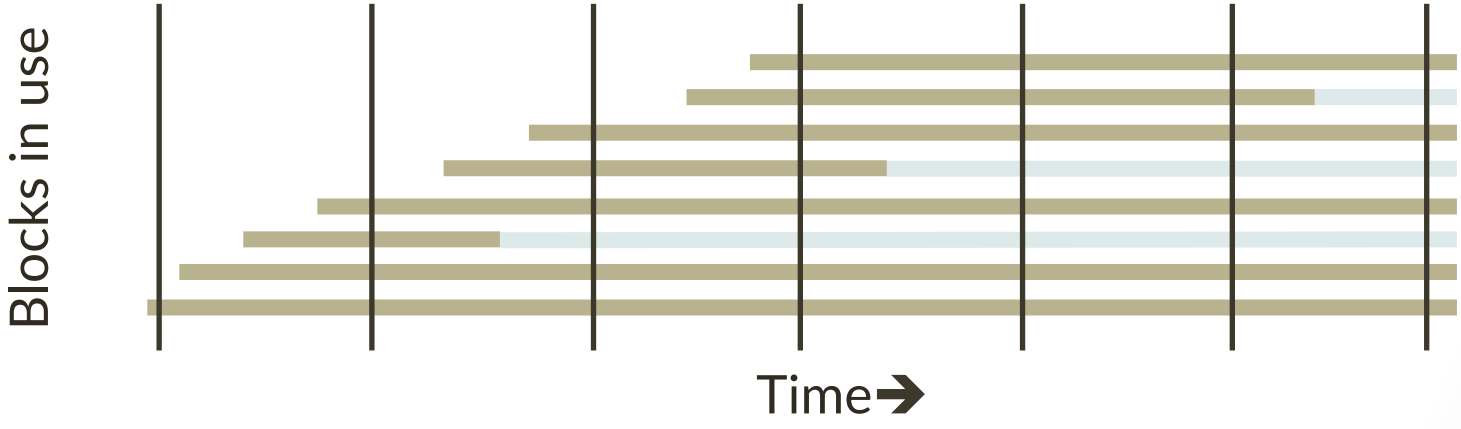
com.delphix:extensible_dataset
com.delphix:bookmarks
com.delphix:enabled_txg
com.delphix:hole_birth

New in OpenZFS – Functional

- Better observability of snapshot size
- Logicalused property (uncompressed used capacity)
- SCSI UNMAP over iSCSI
- libzfs_core – stable thread-safe programming API
- zfs send progress reporting (Bill Pijewski)
- ZFS I/O deadman thread – see if an I/O doesn't complete
- New Dtrace probes – txg states, and generation of errors
- Property to limit number of filesystems/snapshots
- Bookmarks for incremental sends
- zfs list –p (parseable)
- zpool reguid
- Crashdumps can dump to RAIDZ

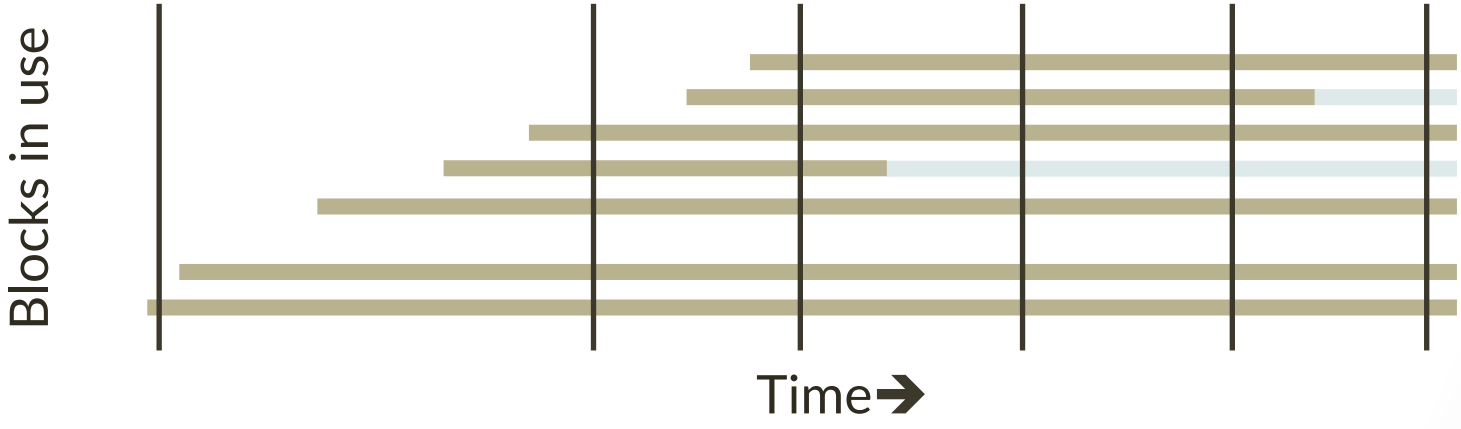
Snapshot size observability

- Assuming you use snapshots…
- If you modify or delete data, you will have some space which is only used by snapshots, not the current dataset.
- You will probably want to recover this eventually.
- Can be hard to identify which snapshots are using large amounts of space.



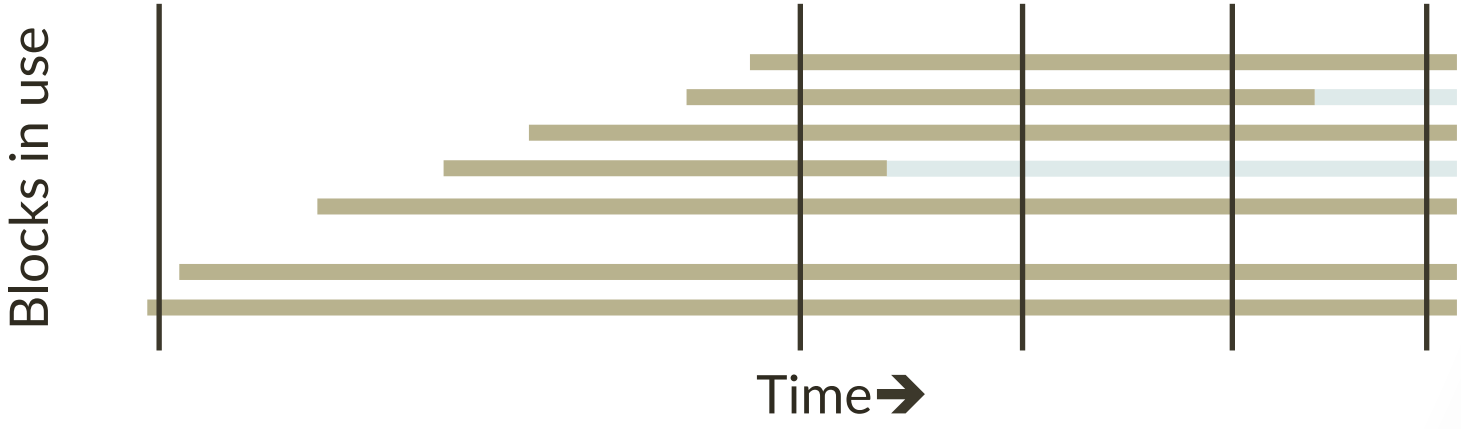
Snapshot size observability

- Assuming you use snapshots…
- If you modify or delete data, you will have some space which is only used by snapshots, not the current dataset.
- You will probably want to recover this eventually.
- Can be hard to identify which snapshots are using large amounts of space.



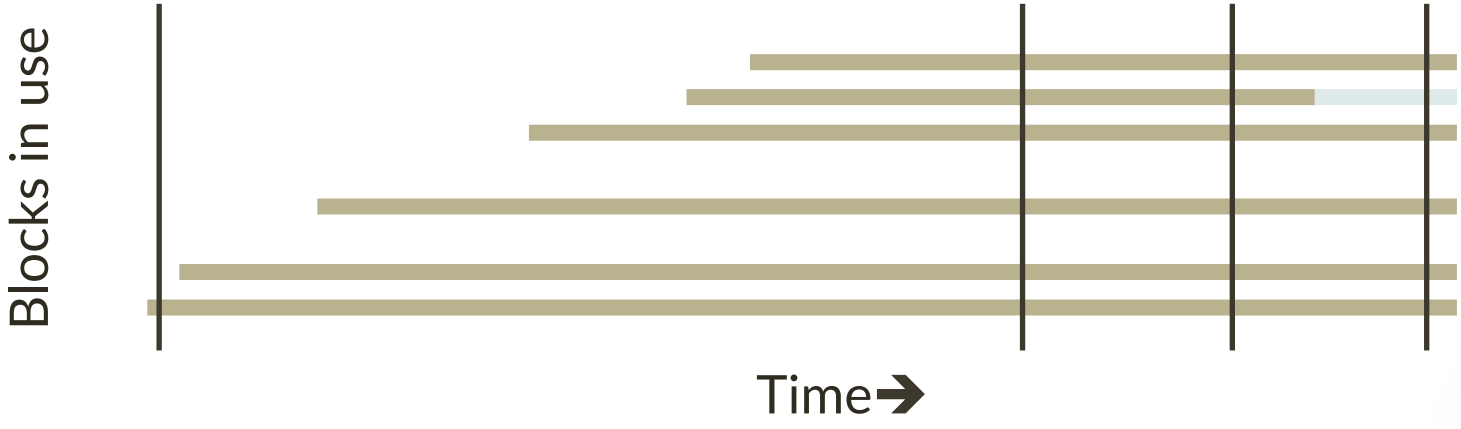
Snapshot size observability

- Assuming you use snapshots…
- If you modify or delete data, you will have some space which is only used by snapshots, not the current dataset.
- You will probably want to recover this eventually.
- Can be hard to identify which snapshots are using large amounts of space.



Snapshot size observability

- Assuming you use snapshots…
- If you modify or delete data, you will have some space which is only used by snapshots, not the current dataset.
- You will probably want to recover this eventually.
- Can be hard to identify which snapshots are using large amounts of space.

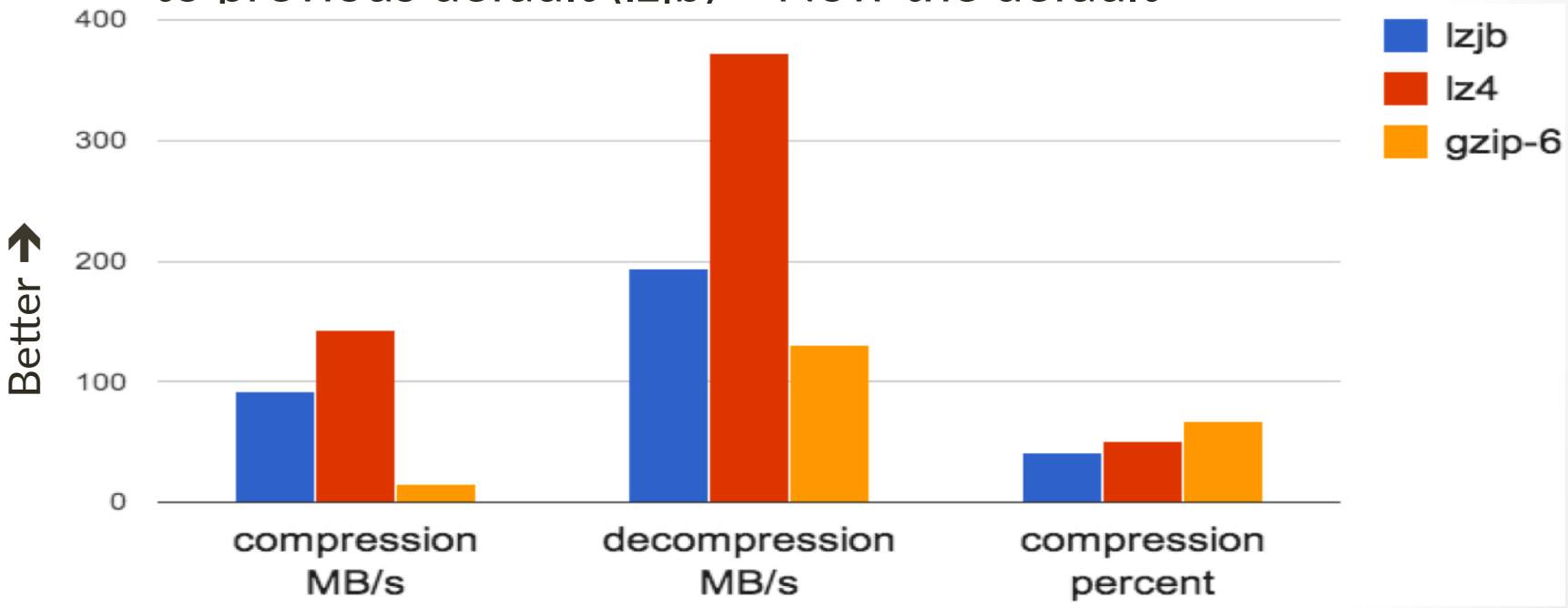


New in OpenZFS - Performance

- Single-copy ARC – don't separately cache snaps/clones
- Separate I/O queues for sync read, sync write, async read, async write, scrub/resilver. Dynamic tuning of queue sizes.
- nop write – don't bother writing back unchanged blocks.
- L2ARC compression – get much more benefit from L2ARC
- L2ARC memory consumption halved
- Metaslabs loaded asynchronously – faster space allocation
- Metaslab histograms – faster to find best metaslab
- Don't send holes multiple times in incremental zfs send streams
- Highly compressed blocks stored in block pointer fields
- redundant_metadata=most
- Large blocksize support (up to 16MiB)
- ARC locking – massive reduction in locking contention

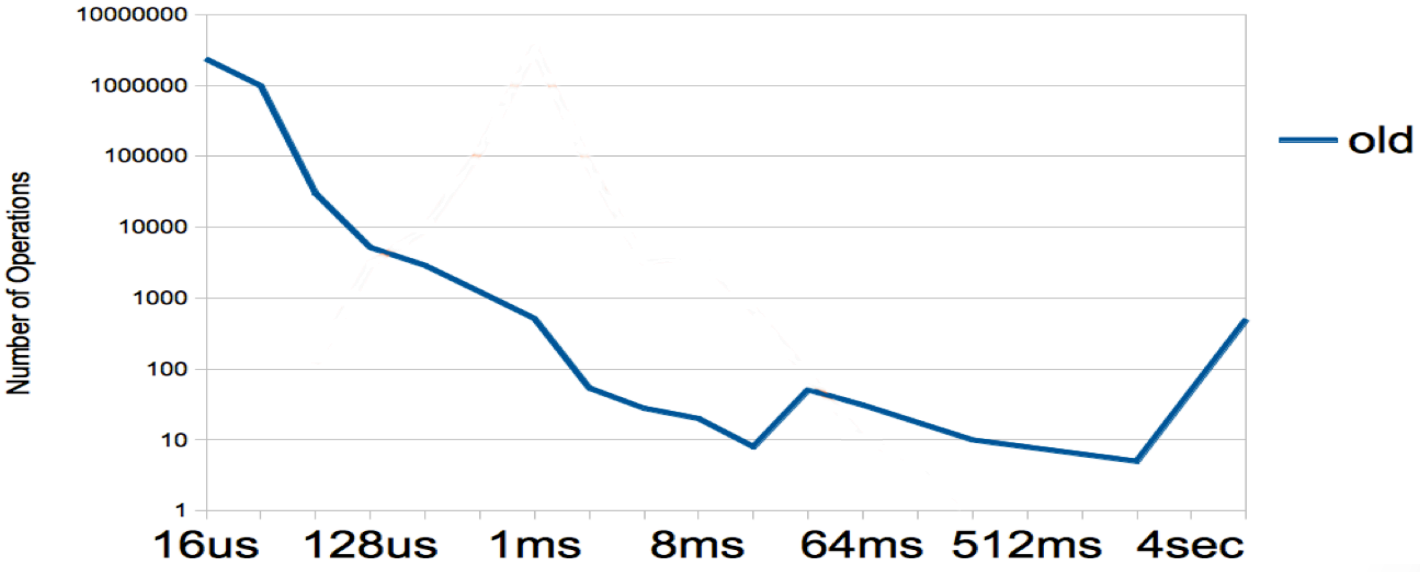
LZ4 compression in OpenZFS

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



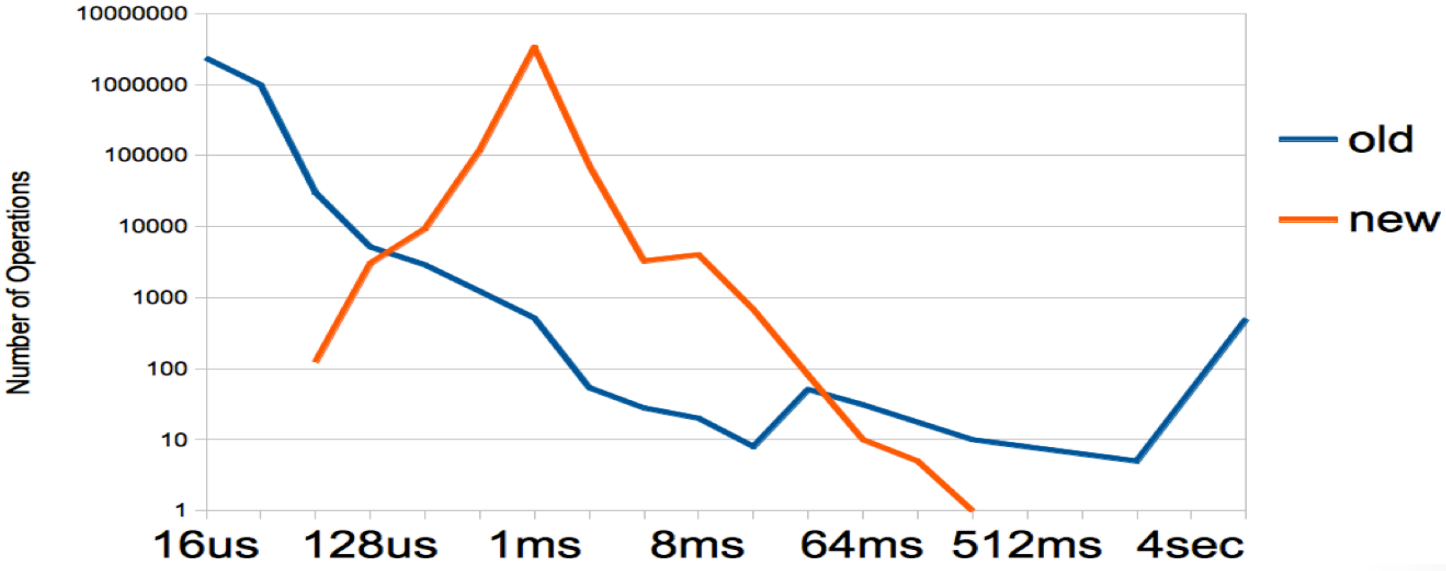
Smoother Write Latency

- 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



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: 5,900 io/s; outliers: 30 milliseconds



Work in Progress in OpenZFS

Complete, pending integration:

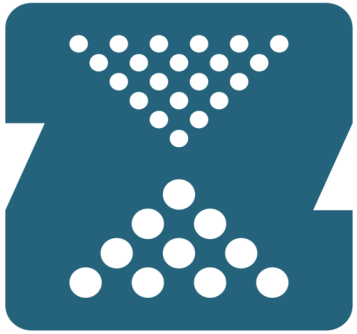
- Prefetch rewrite
- Compressed ARC
- ZFS send/recv performance work (prefetching)
- Resumable send/recv
- Allocation throttle
- Better FMA identification and handling of dying drives

In progress:

- Persistent L2ARC
- Performance optimisation for mixed performance pool disks
- Clone individual files
- Device removal
- Scale Up – more memory (>1TB), more storage (>1PB)

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!
- Form a local group



OpenZFS

Andrew Gabriel
Cucumber Technology Ltd
andrew@cucumber.me.uk