This is a rolling agenda for the monthly OpenZFS Leadership Team Meetings.

If you would like to contribute to this document, please email matt@mahrens.org or zfsleadership@klarasystems.com for access.

Logistics: Meetings are held once every 4 weeks on zoom, Tuesdays from 1:00-2:00pm US Pacific time (¾ of the meetings), or 10-11am PT (¼ of the meetings). The meetings are open to the public, and recordings will be posted on YouTube. Please help us stay on agenda and on time :-)

Zoom Link

Past video recordings

Open Action Items

-

Reach out to @allanjude on slack (or zfsleadership@klarasystems.com) if you would like to be added to the calendar invitation.

Upcoming Agenda Items (for next meeting - 2025-09-09, 13:00 US Pacific)

- https://github.com/openzfs/zfs/pull/17709 - ZTS: Fix fault_limits timeouts
- Build on FreeBSD/i386
  - https://github.com/openzfs/zfs/pull/17708 - Fix atomic-alignment warnings in libspl on FreeBSD/i386
  - https://github.com/openzfs/zfs/pull/17707 - Fix the build of crypto_test on LP32 architectures
  - https://github.com/openzfs/zfs/pull/17706 - Fix the build on 32-bit FreeBSD with GCC
  - https://github.com/openzfs/zfs/pull/17705 - Fix a printf format specifier on FreeBSD/i386
  - https://github.com/openzfs/zfs/pull/17704 - Fix warnings about SHA_2_is_supported on FreeBSD/i386
- https://github.com/openzfs/zfs/pull/17645 - zfs_vnops.c: Add support for the _PC_CLONE_BLKSIZE name
- https://github.com/openzfs/zfs/pull/17630 - added module param to enable decompression of scrubed blocks

- https://github.com/openzfs/zfs/pull/17576 - Add allocation profile export and zleak utility for import
- https://github.com/openzfs/zfs/pull/17573 - Implement new label format for large disks
- https://github.com/openzfs/zfs/pull/17543 - zfs allow send:encrypted
- [Please review] https://github.com/openzfs/zfs/pull/17227 - Detect a slow raidz child during reads
- [Please review] https://github.com/openzfs/zfs/pull/16526 - Multiple pool support for ztest

Agenda Items (2025-08-12, 10:00am US Pacific)

- https://zfsonlinux.topicbox.com/groups/zfs-devel/T60f706738bf1f2e4/openzfs-developer-summit-2025-announcement-and-cfp
- 2.4 Release Schedule
- https://github.com/openzfs/zfs/pull/17567 - Add support for anyraid vdevs
- https://github.com/openzfs/zfs/pull/17543 - zfs allow send:encrypted
- https://github.com/openzfs/zfs/pull/17613 - ZIL make allocations more flexible
- Discuss integrating forced export changes into ztest
- https://github.com/openzfs/zfs/pull/17576 - Add allocation profile export and zleak utility for import
- https://github.com/openzfs/zfs/pull/17625 - zvol: cleanup & fixup zvol destruction sequence and locking
- https://github.com/openzfs/zfs/pull/17573 - Implement new label format for large disks
- https://github.com/openzfs/zfs/pull/17583 - backport enforce arc_dnode_limit to 2.2
- https://github.com/openzfs/zfs/pull/17586 - Add conflict/replacement with older SONAME libzfs and libzpool packages
- https://github.com/openzfs/zfs/pull/17588 - [WIP] Raidz Expansion: multiple devices expanding
- https://github.com/openzfs/zfs/pull/17602 - Fix Assert in dbuf_undirty, which triggers during usage zap shrink
- [Please review] https://github.com/openzfs/zfs/pull/17227 - Detect a slow raidz child during reads
- [Please review] https://github.com/openzfs/zfs/pull/16526 - Multiple pool support for ztest
- https://github.com/openzfs/zfs/pull/17418 - Be more careful with locking db.db_mtx
- https://github.com/openzfs/zfs/pull/16967 - Add native NFSv4 style ZFS ACL support for Linux
-
-

2025-07-15, 13:00 Pacific

- Status update on the Anyraid project
- https://github.com/openzfs/zfs/pull/17543 - zfs allow send:encrypted
- https://github.com/openzfs/zfs/pull/17505 - Allow and prefer special vdevs as ZIL
- https://github.com/openzfs/zfs/pull/17531 - Correct weight recalculation of space-based metaslabs
- https://github.com/openzfs/zfs/pull/17536 - Fix zdb pool/with -k
- https://github.com/openzfs/zfs/pull/17525 - Allow libzpool to set 64-bit values
- https://github.com/openzfs/zfs/pull/17537 - Userspace Tunables
- https://github.com/openzfs/zfs/pull/17388 - zio: add separate pipeline stages for logical IO
- https://github.com/openzfs/zfs/pull/17398 - ZIL: "crash" the ZIL if the pool suspends during fallback
- https://github.com/openzfs/zfs/pull/17227- Detect a slow raidz child during reads
- https://github.com/openzfs/zfs/pull/16835 - zdb raidz file layout
- [Please review] https://github.com/openzfs/zfs/pull/16526 - Multiple pool support for ztest

Previous meetings' notes

2025-06-17, 10:00am Pacific

Video Recording

- Better tracking of what has been backported (can we use git cherry-pick -x to include annotations of the original hashes when merging stuff to stable branches). Standardizing on "Fixes:"
- https://github.com/openzfs/zfs/pull/17463 - Clarify and restrict dmu_tx_assign() errors
- https://github.com/openzfs/zfs/pull/17455 - Default to zfs_bclone_wait_dirty=1
- Investigate if scrub can detect 'there are TXG waiters' and hurry up
- https://github.com/openzfs/zfs/pull/17423 - FreeBSD: Wire up projects support
- https://github.com/openzfs/zfs/pull/17418 - Be more careful with locking db.db_mtx
- https://github.com/openzfs/zfs/pull/17388 - zio: add separate pipeline stages for logical IO
- https://github.com/openzfs/zfs/pull/17398 - ZIL: "crash" the ZIL if the pool suspends during fallback
- [Please review] https://github.com/openzfs/zfs/pull/16853 - Add TXG timestamp database

- [Please review] https://github.com/openzfs/zfs/pull/17004 - Implement dynamic gang header sizes
- [Agree to close?] https://github.com/openzfs/zfs/pull/16579 - libzutil: allow to display powers of 1000 bytes
- Add support for displaying non-byte numbers as units of 1000 not 1024
- [What to do?] https://github.com/openzfs/zfs/pull/17075 - fix ZFS command output uses wrong comma separator
- [Followup and finish] https://github.com/openzfs/zfs/pull/17045 - ZTS: include microsecond timestamps on all output
- https://github.com/openzfs/zfs/pull/16526 - Multiple pool support for ztest
- [Easy for beginner] https://github.com/openzfs/zfs/pull/15658 - zvol_disk_open() may spin on CPU

2025-05-20, 1:00pm Pacific

Video Recording

- Introducing Anyraid (Klara, Eshtek)
    - New vdev type to improve experience of mismatched device sizes
    - Support for mirroring in v1, raidz later
    - Other possibly useful features (e.g. rebalancing, expansion, contraction)
- https://github.com/openzfs/zfs/discussions/17225 ZFS Label Redesign (Klara)
    - Improve the label to handle larger sector sizes, new features
    - Introduced as last years Dev Summit
    - Keep more uberblocks for rewind, possibly grandfathering scheme
    - Interactions with zpool checkpoint, MMP, other things that borrow uberblock slots
- https://github.com/openzfs/zfs/pull/17335 Add synchronous zedlets
- https://github.com/openzfs/zfs/pull/17341 Add hierarchical project quota support (Nutanix)
- https://github.com/openzfs/zfs/discussions/17118 What's the proper way to protect dmu_buf_impl.db.db_data
- https://github.com/openzfs/zfs/pull/17159 Fix inode eviction and sync writeback deadlock on Linux (Nutanix)
- https://github.com/openzfs/zfs/pull/17076 Add a diagnostic kstat for obtaining pool status
- OpenZFS 2.4 release schedule: https://github.com/openzfs/zfs/discussions/17310

2025-04-22, 1:00pm Pacific

Video Recording

- https://github.com/openzfs/zfs/pull/17218 Wire O_DIRECT also to Uncached I/O
- https://github.com/openzfs/zfs/pull/17246 Introduce zfs rewrite subcommand
- https://github.com/openzfs/zfs/pull/17231 Fix double spares for failed vdev
- https://github.com/openzfs/zfs/pull/17193 Add zpool status --lockless|--trylock
- https://github.com/openzfs/zfs/pull/17209 More aggressively assert that db_mtx protects db.db_data

- https://github.com/openzfs/zfs/pull/17214 zfs-2.3.2 patchset
- https://github.com/openzfs/zfs/pull/17227 Detect a slow raidz child during reads
- https://github.com/openzfs/zfs/pull/17159 Fix inode eviction and sync writeback deadlock on Linux

Previous meetings' notes

2025-03-25, 1:00pm Pacific

Video Recording

- https://github.com/openzfs/zfs/pull/17094 Range_tree: Add zfs_recover_rt parameter and extra debug info
- https://github.com/openzfs/zfs/pull/17142 kstat: allow multi-level module names (support for vdev stats queue)
- https://github.com/openzfs/zfs/pull/16205 Hierarchical bandwidth and operations limits (rebased in December)
- https://github.com/openzfs/zfs/pull/16486 Implement parallel ARC eviction and https://github.com/openzfs/zfs/pull/16487 Implement parallel dbuf eviction
- Raidz slow IO
- https://github.com/openzfs/zfs/pull/16967 Add native NFSv4 style ZFS ACL support for Linux
- https://github.com/openzfs/zfs/pull/17004 Implement dynamic gang header sizes
- https://github.com/openzfs/zfs/pull/17073 Make ganging redundancy respect redundant_metadata property
- https://github.com/openzfs/zfs/pull/17111 Implement allocation size ranges and use for gang leaves
- https://github.com/openzfs/zfs/pull/17123 Handle interaction between gang blocks, copies and FDT
- https://github.com/openzfs/zfs/pull/17020 Unified allocation throttling
- https://github.com/openzfs/zfs/pull/17119 Delete dead code: dbuf_loan_arcbuf
- https://github.com/openzfs/zfs/pull/17130 Implement default user/group/project quotas including object default quotas
- https://github.com/openzfs/zfs/pull/17136 Always perform bounds-checking in metaslab-free-concrete
- https://github.com/openzfs/zfs/pull/17143 dmu_tx:rename dmu_tx_assign( ) flags from TXG_* to DMU_TX_* (Ground work for ZIL fail)
- https://github.com/openzfs/zfs/issues/16626 data corruption due to inconsistent locking of db.db_data
- https://github.com/openzfs/zfs/pull/17164 Fix lock reversal on device removal cancel
- *Add your agenda item ideas here and we'll discuss them at the next meeting*

Previous meetings' notes

2025-02-25, 10am Pacific

Video Recording

- zpool fsck (design discussion)
- https://github.com/openzfs/zfs/pull/15215 - slack compression
- https://github.com/openzfs/zfs/pull/17020 - Unified allocation throttling
- https://github.com/openzfs/zfs/pull/16283 - Implement defaultuserquota/defaultgroupquota (UID/GID/ProjID of -1 as default quota key, as alternative to)
- https://github.com/openzfs/zfs/pull/16026 - Add diagnostics kstat for obtaining pool status
- https://github.com/openzfs/zfs/pull/17094 - Revise and extend zfs_recover support for add/remove
- https://github.com/openzfs/zfs/pull/17081 - Better fill empty metaslabs
- https://github.com/openzfs/zfs/pull/17073 - Make ganging redundancy respect redundant_metadata property
- https://github.com/openzfs/zfs/pull/17049 - Add more DDT tests
- https://github.com/openzfs/zfs/pull/17038 - Rework FDT dedup log sync
-
- *Add your agenda item ideas here and we'll discuss them at the next meeting*

Previous meetings' notes

2025-01-28, 1pm Pacific

Video Recording

- Fast Dedup pacing changes
- ZFS wide background work pacing
- https://github.com/openzfs/zfs/pull/16986 - Expand fragmentation table
- https://github.com/openzfs/zfs/pull/16967 - Add native NFSv4 style ZFS ACL support for Linux
- https://github.com/openzfs/zfs/pull/16929 - optimize recv_fix_encryption_hierarchy
- https://github.com/openzfs/zfs/issues/16913 - Allow zpool status -c to also specify vdev properities
- https://github.com/openzfs/zfs/pull/16991 - Force receive permissions and not only.

2025-01-07, 1pm Pacific

Video Recording

- 2.3.0 release expected next week
- Ztest throwing a lot of errors
- Feature proposal: ZFS allow 'send raw only'
- https://github.com/openzfs/zfs/pull/16929 - optimize recv_fix_encryption_hierarchy
- https://github.com/openzfs/zfs/pull/16900 - Detect a slow raidz child during reads
- https://github.com/openzfs/zfs/pull/16835 - zdb raidz file layout
- https://github.com/openzfs/zfs/pull/16817 - Linux: sync mapped data on umount

- https://github.com/openzfs/zfs/pull/16818 - Linux: syncfs(2) should sync all cached files
- https://github.com/openzfs/zfs/pull/16747 - On-demand log-spacemap flush; zpool condense command
- https://github.com/openzfs/zfs/issues/16913 - Allow zpool status -c to also specify vdev properities
- 141 open PRs

2024-12-03, 1pm Pacific

Video Recording

- DirectIO for FreeBSD:
    - Remove an incorrect assertion in zfs_getpages(): https://github.com/openzfs/zfs/pull/16834
    - Enabled by default: https://github.com/openzfs/zfs/pull/16761
- Block cloning prefetcher: https://github.com/openzfs/zfs/pull/16814
- Flush ARC async when exporting: https://github.com/openzfs/zfs/pull/16215
- “Special” allocation class failsafe feature: https://github.com/openzfs/zfs/pull/16185
- Scrub all changes since previous scrub: https://github.com/openzfs/zfs/pull/16301
- RAID-Z file layout: https://github.com/openzfs/zfs/pull/16835
- FreeBSD: support for UF_NOUNLINK https://github.com/openzfs/zfs/pull/16820
- Optimize RAIDZ expansion: https://github.com/openzfs/zfs/pull/16819
- Slack Compression: https://github.com/openzfs/zfs/pull/15215
- Types of special blocks

2024-11-05, 1pm Pacific

Video Recording

- Investigating: zfs ddtprune -p 100 seems to cause corruption: https://github.com/openzfs/zfs/issues/16713
- Use deferred free mechanism for large deletes, to avoid OOM from too-many-ZIOs: https://github.com/openzfs/zfs/pull/16722
- Ztest on multiple pools: https://github.com/openzfs/zfs/pull/16526
- Hold rangelock in zfs_getpages() on FreeBSD for DirectIO: https://github.com/openzfs/zfs/pull/16643
- ZFS send use prefetched spill blocks: https://github.com/openzfs/zfs/pull/16701
- Scrub all changes since previous scrub: https://github.com/openzfs/zfs/pull/16301
- Implement default user/group quotas: https://github.com/openzfs/zfs/pull/16283
- Flush ARC async when exporting: https://github.com/openzfs/zfs/pull/16215
- “Special” allocation class failsafe feature: https://github.com/openzfs/zfs/pull/16185
    - Why can’t this be turned on/off at runtime?
- How does this slightly-less-early time work for everyone? How often should

10/8/2024, 1pm Pacific

Video Recording

- OpenZFS Developer Summit
- Discussed suspend when no redundancy
- Snapshot of critical data (just preserve meta data)

9/10/2024, 1pm pacific

Video Recording

- OpenZFS Developer Summit: submit your talk ideas to Matt by this Friday
- 2.3 release status (Brian)
- ARC async teardown https://github.com/openzfs/zfs/pull/16215 [Klara]
- Parallel ARC eviction https://github.com/openzfs/zfs/pull/16486 [Klara]
- Parallel DBUF eviction (and search improvements) https://github.com/openzfs/zfs/pull/16487 [Klara]
- ZED fault diagnosis [Klara]
- ztest with multiple pools https://github.com/openzfs/zfs/pull/16526 [Klara]
- Fast Dedup is merged [Klara]
- spa_namespace lock improvements https://github.com/openzfs/zfs/pull/16507
- zio_flush: propagate ZIL errors https://github.com/openzfs/zfs/pull/16314 [Klara]
- Ratelimits https://github.com/openzfs/zfs/pull/16205
- Adding userland as a new OS https://github.com/openzfs/zfs/pull/16492 [Klara, pjd]

8/13/2024, 1pm Pacific

Video Recording

- 2.3 release status (Brian)
- Removing legacy support (older FreeBSD and Linux kernels) [Klara]
- Metaslab selection and weighting, especially with large records (16MB) [Klara]
- Metaslab sizing (16GB is not a good maximum with modern pool sizes) [Klara]
- Tunable to disable zfs_readdir prefetch https://github.com/openzfs/zfs/pull/16318
- Fast Dedup status update https://github.com/openzfs/zfs/discussions/15896 [Klara]
- ARC statistics reset (Do we want an interface to reset all stats?) https://github.com/openzfs/zfs/pull/16441
- L2ARC: optionally cache MFU data only (but still MRU/MFU metadata) https://github.com/openzfs/zfs/pull/16402
- Scrub from last completed scrub: https://github.com/openzfs/zfs/pull/16301 [Klara]
- Zpool reguid to a specific GUID: https://github.com/openzfs/zfs/pull/16239 [Klara]
- Export: flush ARC in background: https://github.com/openzfs/zfs/pull/16215 [Klara]
- OpenZFS Developer Summit CFP (Matt)

7/16/2024, 9am Pacific

Video recording

- When should BRT be re-enabled by default?
    - https://github.com/openzfs/zfs/pull/16337 needs review and merge
    - Do we have a specific testing plan for this? I thought that at one point Pawel said he had a large external set of tests for this…
- Should we continue the earlier time meetings?
- 2.3 release schedule
- Status of planning for dev+user summit
- RFC: ZTS: Use QEMU for tests on Linux and FreeBSD (https://github.com/openzfs/zfs/pull/15838) (Tino) - Sample run: https://github.com/openzfs/zfs/actions/runs/9956164411
- *OpenZFS Wiki - a lot users there - is this needed? (https://openzfs.org/w/index.php?title=Special:ListUsers&offset=&limit=500) … maybe remove those, who didn't write anything to the wiki? (Tino)*
- *State of TRIM support in FreeBSD, currently it's not tested within ZTS (explicit exclude via cli_root/zpool_trim/setup.ksh) (Tino)*
- *I really wish that this recurring problem of cannot import pool after a reboot or power failure could be solved in a more user-friendly way than the tip destroy pool and recreate from backup source. If everything is OK, import pool. If not, the owner should eg. starts zpool import -i in interactive mode and zfs then says what it cannot restore and asks whether it should continue without it or whether it should provide read-only data. The previous solutions with -c, -f, -F and -X and then maybe nothing works also are really ugly, although up until the reboot everything was, at least apparently, OK. (PS: I prefere zpool scrub every weekend because it doesn't do any harm.)*
- *https://github.com/openzfs/zfs/pull/16215 (ARC async teardown) (Don Brady)*
- Linux memory pressure update (Linux workaround, ZFS PR)
- Is there a reason channel programs do not support creating clones? Should they?
    - Do people have many success stories about Channel Programs in the wild? I've not heard many, and it seems like significant complexity for a feature I've yet to hear of anyone using deliberately… - Rich E
    - I know Jan B is using them for making bhyve VMs, and someone (I'm drawing a blank on the name) is using it in their snapshot transfer solution (A. Hettinger)
    - I suppose I specifically ask because I know the Lua interpreter causes kASAN to scream every time it's run, and I don't know if anyone wants to do the work to fix that, as I think the patch is nontrivial…
- *https://github.com/openzfs/zfs/pull/16301 - scrub txg*

6/18/2024, 1pm Pacific

Video recording

- Send –no-encryption for -p/-R/etc without preserving encryption https://github.com/openzfs/zfs/pull/15310 (Rich E)
- EL7 goes EOL on June 30 (Tony H.)

- Hierarchical bandwidth limits: https://github.com/openzfs/zfs/pull/16205 (pjd)
- Libzpool - userland ZFS daemon for send/recv (pjd)
    - Deduplicate OS specific code
    - Treat userspace as another supported OS
- Async ARC flush on export: https://github.com/openzfs/zfs/pull/16215 (Klara)
- Dedup Quota: https://github.com/openzfs/zfs/pull/15889 (Klara)
- DDT Preload: https://github.com/openzfs/zfs/pull/15890 (Klara)
- Fast Dedup: https://github.com/openzfs/zfs/pull/15892 (Klara)
- “flat” DDT entry format: https://github.com/openzfs/zfs/pull/15893 (Klara)
- FDT Storage Class: https://github.com/openzfs/zfs/pull/15894 (Klara)
- Dedup log https://github.com/openzfs/zfs/pull/15895 (Klara)
- Dedup prune: https://github.com/openzfs/zfs/pull/16277 (Klara)

5/21/2024, 1pm Pacific

Video recording

- Situation with memory pressure on Linux, especially with Multi-Gen LRU enabled lately (Alexander Motin): see CONFIG_LRU_GEN_ENABLED and set_initial_priority() function in 6.6 LTS kernel and up.
- Hierarchical ratelimits functionality is ready for review (Pawel Dawidek): https://github.com/openzfs/zfs/pull/16205
- ARC Reference counting bug: https://github.com/openzfs/zfs/issues/15802 (Klara)
- Dedup Quota: https://github.com/openzfs/zfs/pull/15889 (Klara)
- DDT Preload: https://github.com/openzfs/zfs/pull/15890 (Klara)
- Fast Dedup: https://github.com/openzfs/zfs/pull/15892 (Klara)
- Taskq stats https://github.com/openzfs/zfs/pull/16171 (Klara)
- Vdev stats https://github.com/openzfs/zfs/pull/16200 (Klara)
- Getting 2.3 in shape for release this fall

4/23/2024, 9am Pacific

Video recording

- ZIO pipeline synchronicity problem (Alexander Motin)
- NULL zios
- OpenZFS Conference
- Fast Dedup Reviews: https://github.com/openzfs/zfs/discussions/15896
    - ZAP shrinking has been accepted
    - Dedup quota needs a 2nd reviewer
- Direct IO PR https://github.com/openzfs/zfs/pull/10018

3/26/2024, 1pm Pacific

Video recording

- As the zfs encryption layer developer isn't there anymore is zfs native encryption in actual state useful for production systems related to version 2.1.15 and 2.2.3 or if not probably when?
- Fast Dedup Reviews: https://github.com/openzfs/zfs/discussions/15896
    - ZAP shrinking has been extensively reviewed by Alexander Motin, need a 2nd reviewer to sign off
- Improvements to zinject: https://github.com/openzfs/zfs/pull/15953
- Single file scrubbing: https://github.com/openzfs/zfs/pull/16018
- JSON kstat for zpool status: https://github.com/openzfs/zfs/pull/16026
- Speculative prefetch for reordered requests: https://github.com/openzfs/zfs/pull/16022
- OpenZFS DevSummit survey for planning the upcoming 2024 conference
- When a program want to start and needs memory the kernel free up memory from used filesystem cache in linux but when you try that with zfs arc will not free memory and the application cannot get its requests, so you have to deal with the arc max kernel parameter before start the app. Will that change anytime in the future ?

2/27/2024, 1pm Pacific

Video recording

- Update ZFSd to match recent new features added to ZED (vdev properties for io/checksum/slow counts that cause device replacement): https://reviews.freebsd.org/D44043
- Fast Dedup is up for review: https://github.com/openzfs/zfs/discussions/15896
- Don't stall MMP thread on slow I/O: https://github.com/openzfs/zfs/pull/15839
- Support snapdir=disabled: https://github.com/openzfs/zfs/pull/15891
- Klara has started work on improving arc_evict() for large memory systems
- FYI, (0) days since last block cloning bug (https://github.com/openzfs/zfs/issues/15933)

1/30/2024, 9am Pacific

Video recording

- OpenZFS 2.2.3 release: https://github.com/openzfs/zfs/pull/15836
- Status of force pool export work: https://github.com/openzfs/zfs/pull/11082
- Status of additional testings via QEMU: https://github.com/openzfs/zfs/pull/15838 (sample: https://github.com/mcmilk/zfs/actions/runs/7709789531) (Tino Reichardt)
- MMP pool suspend from slow disk (Don Brady)
- Do we really still want part9? (Rich Ercolani)

1/2/2024, 1pm Pacific

Video recording

- Block cloning bug recap
- Ubuntu packages for releases (and main branch?)
- Parallel import
- Asynchronous ARC draining
- abd/bio alignment and splitting

12/5/2023, 1pm Pacific

Video recording

- #15526 and related fallout: https://github.com/openzfs/zfs/issues/15526
- Add slow disk diagnosis to ZED https://github.com/openzfs/zfs/pull/15469
- BIO page splitting: https://github.com/openzfs/zfs/pull/15588
- Block cloning tests: https://github.com/openzfs/zfs/pull/15631
- ZIL prediction: https://github.com/openzfs/zfs/pull/15635
- OverlayFS: https://github.com/openzfs/zfs/issues/15581

11/7/2023, 9am Pacific

Video recording

- Add slow disk diagnosis to ZED https://github.com/openzfs/zfs/pull/15469
- Online rollback with mmap()
- arc_prune()
- Arc_evict/dbuf_evict CPU usage
- Integrating ARC with page cache

10/10/2023, 1pm Pacific

Video recording

- Zil performance

9/12/2023, 1pm Pacific

Video recording

- 2.2 release is close to final RC
- Parallel dataset syncing PR
- Fast Dedup design
- Scrub specific txg range design
- Verifying data on pool import
- Flushing various logs to optimize pool import time
- BRT for file concatenation
- “Zfs receive” error handling

8/15/2023, 9am Pacific

Video recording

- Reminder: OpenZFS Developer Summit **October 16-17**. CFP by September 5th
  Email: matt@mahrens.org, including a 1-2 paragraph abstract.
- ZED slow I/O auto-spare [Allan]
- ZED enc-path auto-replace [Allan]
- RAID-Z Expansion [Don]
- Fast Dedup prototype update (with performance comparison) [Allan]
- Per-user reservations [Allan]
- ZIL locking refactor v2 #15122 [Alexander]
- Tunable Compression Threshold #15174 [Rich]

7/18/2023, 1pm Pacific

Video recording

- Raid-Z Expansion https://github.com/openzfs/zfs/pull/15022 (Don)
- Gang ABDs causing alignment issues on 4Kn drives (Allan Jude)
- Review Request: https://github.com/openzfs/zfs/pull/15050 BRT for Linux (Allan Jude)
- ZED Enhancements re: slow I/Os
- Are there any side effects to NOT unloading metaslabs (metaslab_debug_unload)?
- Test failures

6/20/2023, 1pm Pacific

Video recording

- RAIDZ Expansion (Don Brady)
- Add more Github Action runners (arm64, amd64, ppc64 via openstack machines of the Oregon State University Open Source Lab (Tino Reichardt)
- Dedup improvements (Allan Jude)

5/23/2023, 9am Pacific

Video recording

- OZDS dates (Matt)
- ZIL locking improvement (#14841)

4/25/2023, 1pm Pacific

Video recording

- Block cloning status after FreeBSD merge (Pawel Dawidek)
- Hierarchical rate limiting demo (Pawel Dawidek)
- I get in some days / weeks 6x POWER and 6x AARCH VM's for Github action runners (4 Cores, 4GB RAM, 40GB Disk) /Tino
- Update on 2.2 timing perhaps? Lots of cool stuff queued up
- RAIDZ Expansion (anonymous vandal)
- OSX support update
- Direct IO Support

3/28/2023, 1pm Pacific

Video recording

- The OpenZFSonosx developers suggest it is time to decide if it is desired to merge the macOS port. #12110 The PR is currently up to date, and is the latest installable release.
- IO rate limiting (Pawel Dawidek, Allan Jude)

2/28/2023, 9am Pacific

Video recording

- Block Cloning status update (Pawel Dawidek)
- Shared Slog PR introduction (Paul Dagnelie)
    - https://github.com/openzfs/zfs/pull/14520
- Discussed mirrored L2ARC and what it is not allowed, and possibly allowing RAID-Z for special vdevs
- I/O rate limiting properties issues (Pawel / Allan)
- fsync to return error on failmode=continue (Allan Jude)

1/31/2023, 1pm Pacific

Video recording

- Will 2.2 be branched at some point soon? (asking because overlayfs support is not planned to be backported to 2.1.)
- ~~Spurious EIO with zfs-2.1.8 https://github.com/openzfs/zfs/issues/14413~~
- I recently submitted PR#14249, adding Chacha20-Poly1305 as an encryption option. I'd like to talk to someone about next steps towards getting this PR reviewed and merged. (Rob N)
- ZFS Space calculations (raidz deflation)
- More adaptive (data/metadata) ARC eviction: More adaptive ARC eviction. #14359 - openzfs/zfs · GitHub

1/3/2023, 1pm Pacific

Video recording

- I recently submitted PR#14249, adding Chacha20-Poly1305 as an encryption option. I'd like to talk to someone about next steps towards getting this PR reviewed and merged. (Rob N)
- Should the documentation have warnings about using native encryption, similar to dedup? (Rich E)
    - Several people who got burned by native encryption recently asked me why there were no warnings around it if it has known bad failure modes, and I didn't really have a good answer.
    - Klara investigating: https://github.com/openzfs/zfs/issues/12775
- mmap issue: https://github.com/openzfs/zfs/issues/13608
- Curious about the effort to update the ZFS On Disk Format document. Has this been done? If so, where is the latest version?
    - https://github.com/ahrens/zfsondisk

12/6/2022, 9am Pacific

Video recording

- Unmapped prefetch (requesting review) https://github.com/openzfs/zfs/pull/14243
- ZED configuration via vdev properties (requesting review) https://github.com/openzfs/zfs/pull/13805

- Top-level vdev properties (add a ZAP for the root of the pool for property inheritance, for newly added disks)
- Bandwidth quotas

11/8/2022, 9am Pacific

Video recording

- quota performance
- ARC MRU/MFU
- BRT details

10/11/2022, 1pm Pacific

Video recording

- Are we still planning to release 2.2 in November?
    - Would like to see the branch created before the end of this month
- We at iX has grown a need for OverlayFS for Docker purposes, so actively looking on https://github.com/openzfs/zfs/pull/9600 and https://github.com/openzfs/zfs/pull/12209 . The first seems to mostly need Linux VFS expert review.
- Hackathon ideas spreadsheet

9/13/2022, 1pm Pacific

Video recording

- OpenZFS DevSummit talk announcement (Matt)
- How ZFS partitions disks (Michael Dexter)
- Status on encryption bugs (Hendrick)
    - https://github.com/openzfs/openzfs/commit/619c0123a513678d5824d6b1c4d7e8fad3c63e76
    - panic: zfs dbuf code destroys buffer still held by others on encrypted pool
    - https://www.illumos.org/issues/14003
    - https://jira.ixsystems.com/browse/NAS-113491
    - Updating encryption properties on a dataset makes all children unreadable + kernel panic
    - https://github.com/openzfs/zfs/issues/11679
    - https://github.com/openzfs/zfs/issues/12123

8/16/2022, 9am Pacific

Video recording

- ZED configurability: does a vdev property make the most sense? Need 4 new properties
    - Possibly a root-vdev property, or recursive set
- Do we need a failmode=error (or failmode=fail). Where fsync() etc do not hang forever, but actually return an error?

- Discussed investigating changing the default maximum size of a micro-zap
    - Also discussed if the default size of indirect blocks should be lowered (already a tunable)

7/19/2022, 1pm Pacific

Video recording

- OpenZFS conference announcement (Matt)
    - Matt will send out announcements later today
- Relaxed quota enforcement for performance (Allan)
    - When you're close to quota, things get slow.
    - Plan:
        - Change to allow filesystem to go over quota by one txg worth of dirty data or percent of quota, whatever less.
        - For zvols, don't enforce quota at all, or at least allow substantially higher. Quota makes sense for ZVOLs though, at least in some rare scenarios, e.g, thin provisioned ZVOLs.
    - Discussion on whether the behavior should be opt-in via a property or whether we can be a little more relaxed about quotas. Sentiment was that avoiding a property is preferable.
    - Expect a PR soon.
- How best to store configuration for ZED (Allan)
    - Currently, some variables are hard-coded in the ZED C code.
    - It would make sense to make some of these configurable.
    - Question: config file in /etc, or will some command line arguments to ZED suffice?
    - Decision: make it command line options to ZED, then users can use init system features to override arguments.
    - Still, there are use cases for making it a zpool property.
        - Lengthy digression on a weird phantom error that bubbles up to ZED on Linux with certain Broadcom chipsets. For such cases, a zpool property would be appropriate.
- What are the plans for 2.2 release? Or shall I merge scrub performance optimizations into 2.1 branch? (Mav)
    - If we stick to the previous cadence, then the due date for 2.2 is November.
    - Features that are close to completion that we should include if possible
        - Direct IO
        - BRT
- Review requests
    - https://github.com/openzfs/zfs/pull/12773 (spa_asize_inflation) (Allan)
    - https://github.com/openzfs/zfs/pull/9372 (corruption healing recv) (Alek)
- Has anyone experimented with directory scaling (how performance drops off with a large number of files in a directory) (Allan)
    - Allan will provide some data next time.

6/21/2022, 1pm Pacific

Meeting recording

- Give an update on current status of the Direct IO PR *(Brian Atkinson)*
    - Will require the app to not modify memory while write is in progress, but have an opt-in safety check in the ZIO pipeline to catch it. If it is caught, then all of the following happen:
        - ZED event
        - zpool status message
        - write syscall fails with EINVAL
    - Check is off by default because of bcopy+checksum overhead (bcopy is dominant)
    - Marking the page read-only to catch in-flight modification is not an option
    - Brainstorming to avoid overhead:
        - checksum without bcopy (susceptible to transient modification)
        - opportunistic (only check in X% of cases, where X is low enough that performance impact is low)
            - Seems like this option has consensus.
    - Concerns regarding in-flight modification & compression. Checksum might match, but then decompression would fail. So, we need to make sure that compression always copies the buffer, iff it's a userspace buffer (abd_borrow_buf_copy doesn't guarantee copy right now).
    - Also: Brian has one weird failure on FreeBSD. Has a branch with more debug statements & reproducer. Needs help from FreeBSD devs!
- Zio taskq scaling for multiple pools (Allan)
    - zio_taskq_batch_pct defaults to 75% of CPU cores
    - But with N pools, we create N times those threads, which is much more than actually available CPU cores.
    - Brainstorming session:
        - Make such taskqs global
            - Fairness & starvation shouldn't be a problem because the taskqs are FIFO
            - But, fear of deadlocks if taskq operations are interdependent. Are there any?
            - Also: lock contention on taskq locks is amplified by making them global.
    - Conclusion: not an obvious solution, system-wide taskq is worth experimenting. (=> good hackathon project)
- Multiple user-keys for encryption (Jonathan Waldrep)
    - UX proposal: use @ notation on the properties to distinguish different keys (keyformat@key1, keyformat@key2, …)
- Soliciting preferences on #11679 solutions (Rich)
    - NULL pointer deref on encrypted recv
    - It's a race in which you end up taking dbuf_write's arc_write codepath, which temporarily NULLs the buffers of what it's writing, while something else ends up doing a dbuf_read -> … -> arc_buf_untransform_in_place, which unconditionally hands the b_pabd in to decrypt from, and is sometimes NULL unexpectedly.
    - George: if the buffer is being written, it shouldn't be discoverable in the first place.
    - Rich's Summary provides details of how the condition happens.

    - George will take a look to understand why the buffer is discoverable.
- OpenZFS Conference
    - Dates: likely early November.
    - Opportunities to help out are still available (see April 26 meeting)

5/24/2022, 9am Pacific

Meeting recording

- Fiemap[Still needs to address comments from original PR7545] by rohan-puri · Pull Request #9554 · openzfs/zfs · GitHub
- Blake 3
- Block Reference Table
- Write throttle smoothing PR

04/26/2022, 1pm Pacific

Meeting recording

- What about PR #12918 ? Will it be integrated ? (BLAKE3)
    - Looks like it's moving forward.
    - Needs reviews from requested reviewers.
- Matt: Thoughts on the next OpenZFS Conference.
    - High interest in an in-person event among the meeting attendees.
    - Safety measures: decide closer to the conference. But don't relax the guidelines after announcing them.
    - Matt asks for a community member to help with organizing the conference:
        - Work with sponsoring companies
        - Catering
        - AV
        - Venue
        - Date
    - Perk: helping organize means some control. e.g., venue location. The Bay Area would still be a requirement, though.
- Allan: Linux Namespace PR update
    - Will hold fd of the namespace (procfs path) to keep it alive while dataset is jailed. Results in slight UI change: refer to namespace by procfs path.
- Allan: Write Inflation PR
    - Needs help with a failing negative test.
- Michael Dexter: Can OpenZFS native encryption be recommended with confidence or specific caveats, not unlike deduplication?
    - Send/recv has several known bugs.
        - See Rich's spreadsheet (lots of these are likely duplicates of each other)
    - Standalone (= without send/recv) has few bugs.
        - Illumos has an open bug related to projectquota upgrades.

        - Alex Motin: some TrueNAS / FreeNAS users are using native encryption already, plan is to *only* support it, instead of GELI, in future releases.
        - *zfs change-key* has an open issue
    - No crisp answer to the question.
- Rich: update on compression early-abort, using lz4 as heuristic
    - He pushed a bunch of more data.
- BRT: what's the status of the work?
    - GitHub branch was last updates in February
    - Pawel just didn't get around creating the upstream PR. Will try to do that this week.
    - Limitation: with encryption, currently only copies within the same dataset are covered by BRT. We can lift that limitation in a later PR, for datasets in the same clone family.
    - Linux copy_file_range has been hooked up.
    - FreeBSD doesn't have an equivalent yet, so the PR adds a syscall. Long-term, Pawel hopes FreeBSD will grow a generic VOP interface.
- Request for reviews:
    - #13130 support incremental receive of clone streams (DRR_FLAG_CLONE)
    - #12767 fix use-after-free of znode_t in race between zfs_zget and zfs_rmnode/zfs_znode_dmu_fini
    - #9372 corruption correcting recv

03/29/2022, 1pm Pacific

Meeting recording

- Update on compression (Rich)
    - ~~I promise I'll stop eventually~~
    - Design review requested for PR #13244 zstd "early abort"
        - Goal was more or less "make higher zstd levels more generally useful". They currently aren't in scenarios with mixed compressibility data due to high CPU cost on incompressible parts.
        - Approach: use cheap compression algorithms as heuristics to determine whether a large block (>= 128k) is compressible. Only use the high zstd level if heuristic matches. Currently, heuristics are LZ4 first, then zstd-1.
        - Benchmark results:
            - The Pi 4 compression chart on incompressible data is pretty…drastic, going from over 2 hours to 15 minutes to recv a 41 GB dataset at zstd-15. (It's probably even more drastic for 18, but that might take over a day to get the baseline for…)
            - The Ryzen 5900X chart isn't bad either, 4+ minutes to under 1 at zstd-18.
            - Peak losses observed in compression savings were under 100 MB across a 41 GB fairly incompressible and 54 GB very compressible dataset, which seems more than

acceptable to me - and there's an easy tunable if you'd like to not use it.
        - Regarding correctness: Since it doesn't change the compression parameters, it doesn't produce different compressed results, either the same compressed results or uncompressed.
        - Could use reviewers, not that it's a particularly invasive change (I have an even less invasive version, just want to benchmark the additional overhead of it isn't noticeable)
        - Could have a dataset parameter to control whether it's tried or not, if desired
        - Probably also works with LZ4+gzip-1 on gzip, though I haven't tried the experiment yet.
        - Aspects discussed during the meeting that **need follow-up**:
            - Add kstats to keep track of heuristic's decisions. Will help characterize different datasets and determine when this is useful.
            - Need to determine the worst-case CPU-time-saved/compression-lost ratio due to this feature.
                - Synthetic benchmark where we always run lz4, zstd-1, and zstd-4. (zstd-4 is the first level at which this mechanism kicks in.)
                - A pathological dataset where lz4 and zstd-1 heuristic match, but the data is not zstd-4 compressible.
            - Consider publishing the scripts that create the datasets, run the benchmarks, and post-process results.
    - (Below this is just negative results, feel free to skip if people aren't interested in why updating LZ4/zstd/gzip don't look like wins currently)
    - LZ4 update:
        - Branch works, `lz4_version` property lets you swap between 1.9.3 and legacy at will per dataset
        - Want to do more stress testing, add more tests, teach ztest to randomly swap that property while doing its other games, but…
        - Not convinced the compressor's a win, really, performance or savings wise
            - 5900X incompressible chart
            - 5900X compressible chart
        - No feature flag needed, older LZ4 happily decompresses it, even with a version number hidden past where the LZ4 size "header" thinks the stream ends to avoid breaking older readers
        - Send/recv prints an annoying message about not knowing the lz4_version property if you send -p it to someone who doesn't know it, but it doesn't error out
        - PR: If I find a compelling reason to use the new version
    - zstd update
        - Branch works, swapping compressor with `zstd_version` property
        - Plays nice with the above `lz4_version` property too
        - Benchmarking 1.5.0/1.5.1/1.5.2/1.5.0+(SSE|NEON)/1.5.2+(SSE|NEON) continues

            - Preliminary results not very compelling:
                - (“s” versions built with SSE2 optimizations and kernel_fpu dance)
                - 5900X incompressible chart
                - 5900X compressible chart
            - Same thing about not needing a feature flag as LZ4 - though zstd already had a version in its header field, so no fun games needed
            - PR: same as LZ4, if I find a use case it’s an improvement for or a flaw in my approach that results in better performance
        - gzip update
            - Kidding…
            - …but I really did experiment with it.
            - Tried benchmarking a few different zlib forks instead of Linux builtin, none were compellingly different so far
            - Quick and dirty graphs of zlib-ng, which was the most different, here
            - Seems like at best maybe the decompressor might be worth examining, and the SIMD improvements don’t seem to make much difference for our use cases.
- Questions about tests like `refquota_008_neg` that are giving false fails after the write inflation change PR #12773
    - Solution: make the test lower the dirty data max tunable.
- Christian: questions on TBD PR that makes ZTS detect if a test does not properly restore tunable values.
    - test-runner.py is an internal interface, zfs-tests.sh is the external interface of the test suite. Ok to change / cut down test-runner.py functionality that is no longer in use.
    - If we can detect the tunable change, can’t we just reset them and continue running tests? Maybe, if we assume any tunable can be written can be written in any order. But a test that doesn’t restore tunable values properly is a bug. So, just fix any broken tests once in this PR, then this problem should be solved going forward.

03/01/2022, 9am Pacific

Meeting recording

- Update on compression (Rich)
    - Have a ~20% done prototype for modular compressor versions, ended up spending more free time on following up for #13078 first, hope to finish that in the next week or so
    - Have already prepared branches for updated LZ4 (1.9.3) and zstd (1.5.2)
        - Was hoping to wait until lz4 had a new release (they said ETA H1 2022), just to avoid churn or wanting to cherrypick changes since release
        - Want to benchmark 1.5.0/1.5.1/1.5.2, they made a number of performance/compression ratio changes between 1.5.0 and 1.5.1, some of which seem like they might only make sense if we’re also using their SIMD knobs - see 1.5.1’s release notes (and 1.5.0, for

the initial perf improvement mentioning wanting SSE2) to see what I mean.

- Review requests: (Allan)
  - https://github.com/openzfs/zfs/pull/12444 (small, zfskeys rc.d script)
  - https://github.com/openzfs/zfs/pull/12868 (write smoothing)
  - https://github.com/openzfs/zfs/pull/12773 (spa_asize_inflation reduction)
  - https://github.com/openzfs/zfs/pull/12263 (linux namespace delegation support)
- Review requests: (Christian)
  - https://github.com/openzfs/zfs/pull/13130 support incremental receive of clone streams (DRR_FLAG_CLONE)
  - https://github.com/openzfs/zfs/pull/12767 fix use-after-free of znode_t in race between zfs_zget and zfs_rmnode/zfs_znode_dmu_fini

02/01/2022, 1pm Pacific

Meeting recording

- Update on BRT (Block Reference Table) (Pawel)
  - Pawel claims 98% ready.
  - Design changes since last update
    - If a block is already in dedup table, we now bump the dedup refcount instead of starting a second reference counter in the BRT.
    - ZIL: now implemented. Log records just use block pointers. No use of object IDs, since ZIL is per-objset, but BRT operation logged in the ZIL depends on state of source and dest objset.
  - Further potential design changes discussed in the meeting:
    - Have a per-VDEV BRT to avoid repeating the VDEV ID in each entry.
      - => on-disk & memory space savings
  - Discussion about generality of the implementation
    - Current implementation defines two new FreeBSD-specific syscalls
    - Having the Linux copy-reflink IOCTL would make the mechanism “just work” with existing Linux tools.
    - Also, there’s Linux copy_file_range, which “gives filesystems an opportunity to implement "copy acceleration" techniques, such as the use of reflinks.
    - The MacOS syscall is path-based, but Lundman says in chat that the filesystem implements VNOP_CLONEFILE, which operates on vnodes.
    - Ideas about a command line tool / daemon to trigger dedup/BRT offline (i.e., after duplicates were written).
  - Timeline
    - Pawel first needs https://github.com/openzfs/zfs/pull/13027 to be merged.
    - Then he can make a PR.
    - Code will need more reviewers.
- Encryption Bugs (Rich Ercolani)
  - (Was not able to make it to the meeting, summary written ahead of time.)

    - One set of bugs (#12981 et al) got a workaround from George Amanakis - thanks George!
    - Another bug (#12720) manifested as an error during raw send/recv. The underlying cause is faulty on-disk dnodes with contradicting bonuslength/spill pointer flag in < 0.6.4 versions of OpenZFS. Related PR #13014.
    - WIP PR #12943 for issue #11679 had an issue reported, going to try and ameliorate with even more locking, but could use someone with familiarity with send/recv a/o the ARC code to help, as I'm pretty sure this is just papering over something being done incorrectly.
        - Issue #11679 has drawn lots of attention. Someone familiar with the DMU should take a look.
    - WIP I should circulate for extending FORCE_INHERIT/FORCE_NEW_KEY to allow you to escape situations like #12614, need to write more tests, feel free to ping me if you're in this situation and want to try it. (Also trying to figure out what a reasonable thing to do in most cases when you receive a change-key in an incremental send is - so far, all of the options seem to violate POLP sometimes.)
        - Downside: not insignificantly sized foot-gun to allow you to change-key -f
        - Storing last N copies of the wrapped key and trying them all would help you avoid that, but then you have N ways to unlock the key…
    - Someone reported issues with receiving unencrypted under an encrypted and not unlocked parent (#13033; not data loss or anything, just mentioning for completeness)
- Add Blake3 Checksum (PR #12918) (Tino)
    - Are there open issues before inclusion?
    - Benchmarks (Unit is MiB/s, more is better)
    - Is Blake3 a "strong" checksum, in the ZFS sense (= is it dedup-safe)?
        - It is a cryptographic checksum .
        - The patch adds blake3 to the list of supported dedup checksums
- Looking for additional reviews:
    - https://github.com/openzfs/zfs/pull/12773 (reduced spa_inflation)
    - https://github.com/openzfs/zfs/pull/12868 (write smoothing)
    - https://github.com/openzfs/zfs/pull/12263 (linux user namespace support)
    - https://github.com/openzfs/zfs/pull/12789 (log spacemap load time)
    - https://github.com/openzfs/zfs/pull/12767 (use-after free of znode_t)

01/04/2022, 1pm Pacific

Meeting recording

- Encryption Bugs (Rich Ercolani)
    - Rich's spreadsheet is now publicly accessible
    - One new bug was reported that doesn't involve replication: #12931 (dereferencing a wild pointer in the checksum code with encryption+dedup on under heavy load)
- Compressors (Rich Ercolani)

- Issue that sparked the discussion: #12840
- Summary: Replacing the compressor code causes problems if the output is not bit-for-bit identical.
- Problem #1: Uncompressed ARC + Compressed On-Disk + L2ARC
  - Assume uncompressed ARC, which holds the data of a block that is compressed on disk.
  - The L2ARC invariant is that it stores a bit-for-bit identical copy of the on-disk data.
  - So, the L2ARC write must recompress the uncompressed copy.
  - But if the compressor changed, the recompressed copy will not be bit-for-bit identical to the on-disk data, and the L2ARC read will fail with a checksum error, resulting in an L2ARC miss.
- Problem #2: NOP writes
  - TODO explain
- Problem #3: Dedup
  - Hashing is against the compressed blocks. Thus, blocks compressed with the new compressor will not dedup with blocks compressed by the old compressor.
- Initial consensus: all three problematic cases are very minor. We can just replace-upgrade decompressor and compressor.
  - We already did that inadvertently with the different gzip implementations (QAT, Linux, FreeBSD).
  - Uncompressed ARC in general should be considered an edge case. Users should expect that it might not interact perfectly with all other features (such as L2ARC).
  - For L2ARC, we can add a small check that marks recompressed buffers which are not bit-for-bit identical ineligible for L2ARC.
- LZ4 (Rich) and ZSTD are two concrete upgrades that are planned.
- Special treatment of the LZ4 compressor upgrade?
  - LZ4 is the default on many systems.
  - A replacement-upgrade of the LZ4 compressor would thus affect many users of dedup.
  - So, should we require users to opt-in for LZ4 upgrades, e.g., via a feature flag? This would require keeping multiple LZ4 compressor versions around.
  - Decision: Wait for 1 month to see whether someone comes up with a scheme that requires user opt-in. If there is no solution, go ahead with a replacement-upgrade.
- NB: For ZSTD, which changes more frequently, we might not want to maintain multiple compressor versions. So, whatever will be decided for LZ4 does not set a precedent for ZSTD.

12/07/2021, 9am Pacific

Meeting recording

- ZFS Interface for Accelerators (“Z.I.A.”); Jason Lee (Los Alamos National Laboratory)
  - Slides

- Summary: make use of hardware accelerators for compression, checksum, raidz
- Main idea:
  - Data is owned by either the Host CPU or the accelerator.
  - ZIO pipeline refers to data owned by the accelerator through a “handle”.
  - If the accelerator doesn’t support one step of the pipeline, data is moved back to the host for that step. But as long as the next step is supported, we can avoid copying back-and-forth.
  - Discussion about what safeguards should be put in place to keep track of ownership / source-of-truth of the data. E.g., during ABD copy.
- Software architecture provider/consumer of accelerators:
  - “Data Processing Unit Services Module” decouples consumers from providers
  - Non-Restrictive License towards consumers; GPLv2 towards providers
- QA / Discussion
  - Watch out for byte-for-byte compatibility, otherwise incompatible with “uncompressed ARC + L2ARC” use case.
  - Maintainability / Testing: at least a software provider should be part of the OpenZFS repo so that we can prevent accidental breakage (e.g. when changing ZIO pipeline implementation)
  - Public availability of actual hardware & corresponding providers would be critical for upstreaming.
  - Discussion on resource contention (e.g. two zpools sharing one accelerator)
- Native encryption needs some work (Rich Ercolani)
- Spreadsheet of encryption bugs
- Biggest and scariest: incorrect dnode refcounting
  - Panic stacks look very different, trapping at different ASSERTs
  - Not reproducible on x86(_64), but on SPARC and on PowerPC
  - Reproduces within 24 hours on a PowerPC 64 KVM VM (but not qemu-emulated)
- Earliest occurrence dates all the way back to introduction of native encryption
- Other scary category: encrypted send/recv corrupts receive-side dataset
  - One can be triggered by *zfs change-key* => will use wrong wrapping key
- Problem: the community is not aware of a company that could fund fixing this.
- Strategy: bring it up at future meetings to raise awareness.
- Request for reviews:
- Allan: calculate a more realistic allocation inflation multiplier: https://github.com/openzfs/zfs/pull/12773
- Alek: healing recv patch - https://github.com/openzfs/zfs/pull/9372

10/12/2021, 1pm Pacific

Meeting recording

- Block Reference Table update and a demo (Paweł Dawidek)
- A lot of progress has been made and was demonstrated through the demo
- There are still a few interesting problems to be solved specifically:
  - Dedup interaction - Freeing a block that is part of both the BRT and DDT
  - Making the BRT “sendable”
  - Overall memory consumption of feature under certain conditions

- Linux User Namespace Support https://github.com/openzfs/zfs/pull/12263 (Allan Jude)
  - All of the issues in the PR have been addressed
  - The feature should be landing soon
- VDEV Properties https://github.com/openzfs/zfs/pull/11711 (Allan Jude, Mark Maybee)
  - Some of the feedback has been addressed but there is still some progress to be made
- Encryption Incompatible On-Disk Format Change for illumos and OSX - (Jason King)
  - Jason came up with a solution that entails using one of the reserved bits on the object set flag field. The idea is to use that in order to migrate pools over time past this format change.
- OpenZFS Hackathon Ideas
  - Allan brought up the idea of going through our bug backlog and picking bugs that would be nice for people to work on during the hackathon
  - List: https://docs.google.com/spreadsheets/d/1byObqK0clYBpgJfzPUNWKMeHkiUAuFkRbTK71K8eW9g/edit?usp=sharing

9/14/2021, 9am Pacific

Meeting recording

- OpenZFS Dev Summit: talks & schedule (Matt)
  - The talks and other Info can be found in the summit's website: https://openzfs.org/wiki/OpenZFS_Developer_Summit_2021
  - Like previous years we will be streaming the event live on Youtube and provide the recordings there for future reference.
  - There will be a hackathon on the second day as usual. This time it would be great if people from this meeting and the general leadership could lead some of the hackathon efforts. One piece of feedback that we got from previous years was that it would be nice to have people available to help newcomers get started.
  - https://github.com/google/triage-party
- Speed up zfs list (ready for review)
  - PR: https://github.com/openzfs/zfs/pull/11080
  - Status: Looking for reviewers
  - This work adds more fields to an existing optimization where we fetch specific fields from datasets without actually opening their objsets (object sets), which leads to a 300%+ performance improvement for those scenarios.
  - Future ideas/work: Would be for zfs destroy to take advantage of the same optimization and just print the datasets to be destroyed in no-op scenarios without actually calculating the space freed, which is expensive.
- ZFS allow for 'encrypted' send only, to only allow encrypted sends
  - Question for this feature was brought up in FreeBSD podcast last week
  - The idea is that the receiving system of such a send-receive stream won't be able to access/read the plain text data of the encrypted dataset.
  - This level of security is not currently supported and probably involves some changes to be designed outside of the ZFS kernel module.

- Could be a good hackathon topic for the summit!
- ZTS may fail in weird ways when re-ran on the same disks
  - Igor has been experiencing ZTS issues in DilOS where the first time the ZTS runs successfully but subsequent runs experience sporadic failures. His hypothesis so far is that the labels are not cleared for the tests that run on disk-based pools (as opposed to file-based ones).
  - Historically, we've seen that all test cleanups work when tests pass, but sometimes they do experience issues when tests fail. Proper error-handling and cleanup for all possible scenarios during a test failure can get tricky.
  - Igor will be filling an issue after he comes up with a minimal reproducer and/or a list of the troublesome tests so we can test whether the same issues come up in the OS's officially supported by the OpenZFS repo.
- Issue Triaging & Stalebot
  - There exist some concerns within the community about Issue Triaging and the use of the Stalebot to mark and close them.
  - Illumos has been occasionally hitting a panic described in https://www.illumos.org/issues/14003 , which apparently was also reported in OpenZFS under https://github.com/openzfs/zfs/issues/6881 once but was subsequently closed by Stalebot.
  - The underlying problem seems to be issue involvement and triaging from the community. There's recently been efforts from the leadership to assign specific folks to help with PRs (ensuring that they get the attention of the right reviews and people to test them) but unfortunately issue triaging is something that people do on a volunteer-basis. In the meantime the stalebot mark old issues as Stale and if there is no activity on them for 90 days, only then closes them.
  - The next steps on this would be to start a discussion on the issue and encourage hackathon ideas around it during the summit.

8/17/2021, 1pm Pacific

Meeting recording

- OpenZFS call for presentations (matt)

7/20/2021, 9am Pacific

Meeting recording

- OpenZFS Dev Summit: Dates, planning, etc.
  - Registration for talk proposals opens on July 23rd and deadline is on August 30th
  - Conference Nov 8-9th
  - Remote or hybrid (in-person and remote) conference is being discussed. Would folks be interested in the hybrid option? Only Allan expressed interest. Most interest is in remote-only.
- Run ztest with asan (Address Sanitizer malloc debugger) (Matt/Mark)
  There was some initial issue with ztest, issue 12215 that needs to be tracked down before we can enable ASan
  Related: MarkJ@FreeBSD also using KMSAN:
  https://github.com/openzfs/zfs/pull/12383
- Status update on Panzura's temporal dedup?

(matt:) looks like the last update was 4/28/2020, they have “de-prioritized” this upstreaming work

No one on the call has update.

- Allan Jude:
  - Linux Namespace support PR12263 is ready for review
  - About time to remove WIP status for vdev properties PR11711, new tests are being written.
  - Klara is working on zvol performance, if you have known pathologies, please share
- Allan Jude brought up enabling LZ4 compression by default.
- Ryan Moller suggests changing default for xattr to SA
- Alexander Motin suggests changing default zvol block size from 8kb to 16kb
  - Discussed possibly forcing allocations to be far apart on the disk (ala ditto blocks) to improve flash performance
- Xattr compat (Ryan): https://github.com/openzfs/zfs/pull/11919
  Requesting design/code review. Tony to reach out to Brian.
- Formalize a procedure for ‘erratas’, like the encryption+ project dnode issue, or even the zstd big-endian issue (like a bug@ not a feature@ flag)
- Due to zfs projects being ported after encryption, #7177 (include the project dnode as part of the objset hash on encrypted datasets) was missed (this appears to have happened with the macOS port as well). As a result, there are now encrypted zfs datasets on illumos that are incompatible with OpenZFS and vice versa. While it is possible to re-try any objset hash failures for compatibility between the two, ‘fixing’ such a dataset presents some thorny issues. One approach that was floated in the illumos community (see https://illumos.topicbox.com/groups/developer/T1981ce734a4a8c50-M4a3b3c0aff56de6ac0ceddfc) was to propose an SPA feature that’s largely a dummy feature on non-illumos platforms (i.e. is ignored), which would allow for explicit identification of the two pools.

6/22/2021, 9am Pacific

Meeting recording

- Whatever happened with ztour? Is anyone working on it or an equivalent? (Rich)
  - ZTOUR was a FUSE project with ZDB-like functionality. It would mount a pseudo filesystem where one can look at on-disk metadata through files and directories in it.
  - It was started by Don Brady and demoed during the hackathon of a ZFS Developer Summit a few years back.
  - No one is currently working on it and there don’t seem to be any alternatives.
  - People in general think it is a good idea so if anyone is interested in picking that up let Matt know and he can connect them with Don, or just bring Don to the next Leadership meeting.
  - Aside: Something like ZTOUR may be needed for the object-storage project (see below) for exploring the backend’s state.
- Linux User Namespace Support https://github.com/openzfs/zfs/pull/12263
  - First PR is out and everything seems to be in order.
  - The commit implements INGLOBALZONE() for Linux and it tries to map all the jail (FreeBSD)/zone(Solaris/illumos) code to their respective Linux counterparts.

- Allan is interested in reviews but more importantly wants to make sure that people are aware of this change and they get to double-check that it doesn't break any of their existing workflows.
- As a future goal we may want to add additional testing for the jails code too since we are here. The ZTS code had(has?) some conditional logic to ensure that most of its tests pass in a local zone in Solaris/illumos. We may want to use that code as a reference point and, once verified, change it appropriately for jails and Linux containers.

- ZREPL bugs
- There have been a couple of recurring bug reports that stem from zfs send and we have a reproducing dataset for one of them.
- The best way forward would be to file an official bug so others are also aware of it and ping Paul Dagnelie.

- Update on ZFS-on-object-storage (Matt)
- Overview:
  - We want ZFS to be able to consume object storage (currently only block storage is supported). The idea here is to use something like S3 for backend storage instead of "traditional disks/drives". This can be cost efficient especially in cloud environments like AWS where S3 is a quarter or fifth of the cost of EBS.
- Update:
  - We are currently on our way to an initial implementation and most of the Delphix ZFS developers are slowly getting involved in the development of this project.
  - We've also opened an issue in the repo online where we give an overview of the project and we showcase the preliminary user interface in the CLI. Feedback from the community is welcome!
- Other Design Details - User Agent & Zettacache:
  - The general use-case within Delphix is databases which means random writes with small record sizes (as opposed to backups where one can use a big record size and never look back). For this reason we want to combine a lot of small blocks when sending them to the block storage. This way we can ignore the latencies of the backend. For writes that can work well as they can be batched from our side (in the ZIL can be used to persist the writes closer in the access hierarchy).
  - Reads unfortunately are a different story as batching them is not as straightforward and, and waiting for dozens of milliseconds is too slow. We considered using the L2ARC to alleviate this but, besides the fact that it can take up too much memory on the system, its eviction policy is not great and that in turn affects our hit rate.
  - For that reason we have an agent that handles all the object store I/O - including the batching and the caching of data locally. The agent runs in the userland and it is written in Rust.

5/25/2021, 1pm Pacific

Meeting recording

- New maintainers (Matt)

- Brian has been the sole maintainer for ZoL -> OpenZFS for almost a decade now.
- The volume of contribution and users has been increasing and we need more people
- Matt and Tony Hutter will be joining him as maintainers
- Tony Nguyen, John Kennedy, and Mark Maybee will also be helping out by ushering incoming PRs through the review process (e.g. finding reviewers, applying the appropriate tags, and merging them).

- Settings vdev properties at pool create time (Allan)
- This is not a pressing matter for getting the vdev properties PR merged, but if anyone comes up with a good idea on how the argument parsing should be done for this, they should definitely send an email to Allan or open a discussion in the mailing list.

- Getting zpool status details via pool properties (scrub progress, etc) (related issue) (Allan)
- Some of these properties you can get with the vdev properties PR as it is right now but certain pool properties like a scan estimate or other things that admins see from commands like zpool-status are harder to get.
- It would be useful to expose some of these properties through zpool-get as-is or at least parts of them.

- If we add a bunch more pool and dataset properties, do we need a new 'verbose' type or something, where a lot of these new (not for human consumption) properties are not displayed by default?
- zpool/zfs get all has been pretty verbose in general as more properties are piling up on it and its output is not very friendly in terms of parsing it programmatically
- Allan and Matt are considering a potential alternative command option that's more human readable and outputs some important properties (not all of them).

- Userland zfs (PR) (Mayank)
- The above PR needs reviewers
- There is a lot more work to be scoped and done. Ideally the upstreaming work from the cStor folks would involve incremental PRs adding more IOCTLs or other minimal functionality that can be tested independently in the userland and slowly become part of the existing testing
    - Allan brought up an old idea of the userland send/receive tests
    - Other options include but are not limited to Matt's old socket-based userland ZFS testing.
- Keeping in mind FUSE and laying the groundwork for it during this effort should also provide some long-term benefits and flexibility as we discover more ways on how to bring the kernel code to the userland.
- There is a lot of ground to be covered but given the excitement there should be volunteers to help. If you are interested please reach out to Mayank, Matt, or Serapheim.

- Corrective Recv PR (Alek)
- The above PR seems to be passing our tests and needs reviewers/reviews.

- FreeBSD Summit
- Matt will be giving a talk about RAIDZ expansion
- There will be other storage topics covered outside of ZFS for folks that are interested

- ZFS support for LXD and similar container technologies (Allan)

    - A PR should be out soon (probably next week)
    - There will be a need for both reviewers and testers
- ZFS Forced Export (Allan)
    - Tests are passing and a PR should be open soon
- OSX port (Jorgen)
    - Around 71% of the tests currently pass and we'd like to start opening PRs soon
    - Besides the above, one thing that is pending in terms of infrastructure is figuring out which solution to use for incorporating testing for the new OS in our CI/CD. Github Action seems to provide some MacOS instances but there may be a timeout that's not enough for us to run ZTS. AWS is another option.
- Add Blake3 Checksum (PR) (Tino)
    - Sorry, did not have time for this currently (will do it in the next 4 weeks)
    - I got an account on the gcc farm project, so assembler routines for sha2 will also come from me (aarch64, ppc, x64) … together with blake3 and benchmarks
    - (Moving as an agenda item for next time)

4/27/2021, 1pm Pacific

Meeting recording

- Corrective Recv PR (Alek)
    - The functionality introduced here is for being able to "heal" a corrupted dataset from a send file (instead of deleting and recreating it). An interesting detail is that this feature works regardless of the differences in the configuration of the target machine and the machine that generated the test file (e.g. compression or other properties).
    - Current status is that most of the tests are passing except a couple from the redacted send-receive tests in FreeBSD which cause a panic.
        - John Kennedy and Paul Dagnelie can help with this
    - Besides fixing the panic - the PR needs reviews.
- Add Blake3 Checksum (PR) (Tino)
    - Currently experiencing problems with supporting this on FreeBSD. The complications seem specific to the code organization and getting the right directives in autoconf/automake and make to work.
    - Besides just adding this new checksum algorithm, the community would like to see how this algorithm compares to existing ones, running on a sufficiently modern CPU under different block sizes.
- Should I add sha_ni (x86) and maybe also aarch64 sha2 hardware support?
    - There is probably some low-hanging fruit for performance improvements, especially for our choice of CPU instructions.
    - Some support for this in FreeBSD has started being worked on: https://github.com/freebsd/freebsd-src/compare/main...allanjude:zfssha
- Interface for /proc/spl/kstat/zfs/hashing_bench ? echo blake3-sse2 > hashing ..
- ZFS and Linux Containers (LXD) (Allan Jude)
    - The idea here is jailing a dataset (group of datasets) to a user namespace in LXD.
        - In terms of code, this would mean at least implementing the IS_GLOBAL_ZONE() macro that currently in Linux always returns true.
    - Do people have interesting use/test cases?

        - If you do, please do reach out to Allan.
    - What makes sense for extending the ZTS to cover containers?
        - Besides the actual ZTS-specific additions, we'll need to look at what we can reasonably add to our automated testing bots in Github to run those tests.
- Cross-platform compatible xattr support (Alexander Motin - amotin)
    - Current status: There is progress but we'd love it if more people are able to look at the code and/or bring alternative solutions/ideas to the discussion.
- ZFS on Object Storage
    - Currently an experiment within Delphix
    - The idea is to run ZFS on top of an object store like S3 or MinIO.
    - The focus is on performance for small record sizes.
    - There are currently two brand-new components being introduced:
        - A userland agent that facilitates the interaction between ZFS and the object store
        - A new block-based local cache introduced as a vdev type to improve the performance of the overall design of this project.
            - The L2ARC was briefly considered but it doesn't satisfy the requirements for the current use-case as its memory requirements are high.
    - More technical details on the upcoming OpenZFS Summit.
    - Until then, while the technical requirements are still being designed we'd love any feedback on the administrative/user interface of this project.
        - Currently we just provide a couple of user properties just to test the design (e.g. a property for the s3 URL, credentials file, etc..). Eventually if we are to start supporting other protocols we'll need to have a user interface and workflow that works for all of them.

3/20/2021, 9am Pacific

Meeting recording

- Version numbering scheme (Bryan)
    - Major.Minor.Patch
        - Major - support for new platforms and new feature that target very common use-cases
        - Minor - new features, properties, feature flags (e.g draid), refactors
        - Patch - support for new kernels, bug fixes
    - Periodical Releases and LTS
        - We'll be creating a new release branch about every 6 months, marking the occasional LTS. The LTS releases will contain only very critical bug fixes (e.g. security or data corruptions - crashes for rarely used code paths are not part of this).
- OpenZFS Developer Summit (Matt)
    - Will be held online November 3rd-4th Wed-Thu
    - Submit Proposals - Deadline: End of August
        - What are you working on now or the next few months?
        - Ideas: New features, performance investigations/analyses, interesting use-cases/applications/workloads.
    - Will be sending announcements to the mailing list and other channels soon

    - COVID & Travelling: Depending on how things are by that time we may reconsider allowing folks to gather in person somewhere. That said we plan to remain “remote-first” regardless of the situation (e.g. if you want to participate/present but don’t feel safe travelling you can still participate/present remotely)
- Patches to fix bugs and upstreaming from DilOS (Igor K)
    - Igor found some bugs in the dRAID and L2ARC features, for some of which he has fixes for. He’ll be working with the upstream community to create issues for them.
    - He also wants to make integration with DilOS easier. The proper plan for this would be like adding support for any other platform (similar to the MacOS effort taking place currently).
        - This involves things like refactoring code and re-organizing files and folders so we end up with the least amount of ifdefs, more common code among platforms, and any platform-specific code that we cannot avoid to be refactored into its own files.
- RAIDz Expansion (Matt)
    - Currently cleaning up the code and polishing the design doc
    - A PR should be out soon
- Proposal: Optionally calculate the space deleted in zfs destroy -v (Allan)
    - Motivation: For pools with a lot of snapshots the overhead of the space calculation that happens by default incurs a lot of overhead.
    - The idea is not to change the default behavior but at least provide an option to get the list of the snapshots to be deleted without the space calculation
- Performance Improvements for zfs list (Allan)
    - PR: https://github.com/openzfs/zfs/pull/11080
    - The idea here is to improve the performance of zfs list by performing the least computation that is required to gather the specific fields/properties specified by the -o flag.

3/2/2021, 1pm Pacific

Meeting recording

- Vdev properties (Allan)
    - Allan will be posting a PR about this soon to demonstrate the state that the code is currently in. In terms of functionality that PR is expected to work only with read-only properties for the time-being.
    - The UI/Syntax component of this change is currently modelled after other existing zpool commands (e.g. zpool list, zpool iostat, etc..).
    - Mark Maybee will be working on the vdev_noalloc feature and could potentially leverage the vdev properties infrastructure for his change. Depending on the state of the PR he may help Allan push it through the finish line.
- More meaningful error messages for admins using zpool import
    - The idea here was to propagate the debug statements produced during pool import to the userland in order to replace generic import errors that are not particularly helpful. Could be a good hackathon project.
- Cleanup and refactoring of platform independent ZVOL code (Christian)
    - Relevant PR: FreeBSD struct bio support for zfs_uio_t (needed by https://github.com/openzfs/zfs/pull/11657#issuecomment-788860549)

    - Currently having problems dealing with GEOM-related code - Sean Fagan volunteered to help over Slack.
- Pool user properties: https://github.com/openzfs/zfs/pull/11680
- Next Meeting Preview:
    - WIll take place earlier (9am Pacific in 4 weeks)
    - Brian may have some information/updates in the version numbering scheme for OpenZFS
    - Matt may have information/updates for this year's OpenZFS summit.

2/2/2021, 1pm Pacific

Meeting recording

- Seagate command priority call tomorrow
    - Mentioned in the last OpenZFS meeting
    - Get in contact with Mohammed if interested
    - WIll take place tomorrow 11:30 PST
- OpenZFS Versioning Scheme and what it means
    - Motivation: Folks were interested in learning when would dRAID be included in an official release
    - Brian will put something together for the next meeting to be discussed
- Status of specific PRs:
    - DirectIO
        - Brian Atkinson is still working on it. We expect it to be updated soon, at which point we'll need reviewers
    - Pool Feature Compatibility Options during pool creation
        - Still ongoing, needs reviewers
    - Force Pool Export
        - Should be good to go soon. We'd like to get more reviewers as the PR touches many different parts of the codebase.

1/5/2021, 9am Pacific

Meeting recording

- What is on the current roadmap in terms of OpenZFS releases?
    - 2.0.1 and subsequent 2.0.X releases are coming out soon that better support the latest Linux kernel versions.
    - 2.1 is expected to be released in this quarter
    - Main projects that are inflight that we hope to release in 2.1:
        - DRAID, Forced Pool Export, DirectIO
    - Further quarterly releases are expected to mostly have bug fixes
    - We expect OpenZFS 3.0 within the year~18 months (first half of 2022)
- RAIDZ Expansion
    - Work is still underway, expect to see progress on that soon
    - Currently Matt is breaking out some ABD related changes from his branch and a PR with them is expected to be out later this week. These changes mostly make it

possible for the abd_t to be allocated by the caller. The BSD-specific parts of this work are a bit more tricky but still doable.

- Marking vdevs as non-allocatable
    - Mark Maybee is expected be working on this feature
    - Goal: Making the removal of multiple devices easier
        - Ensure that all devices can be removed (e.g. there is enough space in the pool for the data after their removal)
        - Don't spread data of device being removed to devices that will be removed next, mark them as non-allocatable first as you queue up their removal.
    - Current design just passivates the vdev's metaslab group
    - Christian also threw the idea of removing the vdev from the metaslab class altogether, instead of just passivating the group
    - Allan Jude has some code for the vdev-properties he could try and upstream soon
        - No-alloc could be a part of this, and save Mark from having to write an interface to the userland for it
- Potential Talks with Seagate SCSI/Firmware Folks about priority policies
    - Seagate Firmware Folks want to get a better understanding of how ZFS prioritizes operations, and coordinate with the community on their designs of how operations can be prioritized best.
    - From our community's perspective, getting a high-level description of what the driver can do, how we can take advantage of that, and if the benefit is enough for us to implement any logic, are the main questions.
    - Alexander Motin's work/discussion of separating priorities for different requests (ZFS background work, user-triggered requests, synchronous vs asynchronous, etc..) is also relevant on the matter.
    - For next steps on this:
        - It would be nice to have some information from the firmware folks before setting a meeting, so we can have a better understanding of what this entails and gather up the right folks that can discuss the topic in more detail.
        - For setting up the actual meeting, we can coordinate on Slack, have an initial discussion there, and then send out an email when everyone is prepared to talk about this.

12/8/2020, 1pm Pacific

Meeting recording

- OpenZFS 2.0 released!
- DRAID integrated to master!
- Further review requested for forced-export https://github.com/openzfs/zfs/pull/11082
- RAIDZ expansion performance
- Visibility of .zfs/snapshot inside a FreeBSD jail

11/10/2020, 1pm Pacific

Meeting recording

- OpenZFS 2.0 release schedule (Brian)
    - Release expected to come out this month

    - Most probably any recent change from master that is not big commit or a project will be part of it
- DRAID PR (Brian)
    - Expect to be merged this week
- Review request: Forced Export PR (Allan)
    - Main use-case reminder: On multi-pool setups, the user should be able to remove a suspended pool from the system (even if that pool is holding the spa_namespace_lock and thus affecting the operations of other pools).
    - Commit now passes all tests and it is ready to go
- Testing request: RAIDZ Expansion (Matt)
    - Expectation is to have the final PR out around the of this year/ beginning of the next one
    - Would love to see people trying out the current PR to uncover any potential issues/bugs in it.
- Non-interactive zio's (Alexander)
    - Problem: Current prioritization of ZFS internal/background work (e.g. scrubs, trims, etc..) vs priorities of sync and async I/Os coming from the userland (that's what is meant above by the term interactive I/Os - non-interactive being the I/Os from ZFS background work), can hurt performance.
    - Alexander has a patch out that helps improve the overall latencies for certain workloads.
    - There is still a lot of discussion to be made as a couple key questions were raised during the meeting:
        - What's the benefit of increasing the write queue depth when the write-cache is enabled?
        - Why is the limit 10 and not 3?
    - From the general discussion a few key points where made:
        - It does make sense to address background activities from normal I/O separately
        - We should revisit a lot of these parts of ZFS and the assumptions behind them
        - Ensuring that we prevent starvation of any queue should be a good starting point for this exploration
        - Allowing the ability to change different queue depths would be nice. Changing the defaults though would require more upfront experimentation. It would be interesting to also explore if we could change these depths dynamically somehow.

10/13/2020, 9am Pacific

Meeting recording

- Determine AI's from conference discussions
    - Code reviews for Add "features" property for zpool feature sets. · PR #10980
        - PR needs reviewers of 2 kinds:
            - [1] Reviewers of the actual code
            - [2] Reviewers for the feature sets outlined
    - Draid code reviews (Brian)

        - There is a PR out and we are looking for reviewers.
        - Brian is expected to squash his commits today or tomorrow to make reviews easier
        - It is a lot of code so the more reviewers the better - if you are interested in how dRaid works, how it interacts with the rest of ZFS's features, or just want to read the documentation and see the tradeoffs involved in its usage you are more than welcome to take a look at the PR.
        - Brian is also welcoming people to pull the code and test it
    - Block reference tracking design doc (Matt/Pawel)
        - There was some work on writing up an initial design doc in the conference
        - The goal here is to complete the doc and share it in one of the future meetings.
    - Terminology review, make sure we use the terms "Healing resilver" in the docs, and maybe even zpool status etc (Allan)
        - After Mark Maybee's sequential rebuild talk during the summit, it would be nice to have distinct terminology for the two resilver options that we provide.
        - This work entails ensuring that there is no ambiguous terminology used in docs, man pages, and command output.
    - ZDB --cp upstreaming (Allan)
        - Initial PR is out - the span of this work does not include the zdb long-opts refactoring
        - Current status:
            - Need to add some test cases for the feature and add its option to the help message
            - Update PR with misc fixes like the dump_all option bug
        - Potential future work for this would be to use the dnode's blocksize (instead of the standard 1MB used now)
    - Forced Export Feature (Allan)
        - Feature was rebased to latest master with some new improvements
        - PR should be out soon
        - Of particular interest may be the code introduced for dataset unmounting in Linux to work around some specific VFS cases
    - zfs list improvements (Allan)
        - There is some effort in the works of making the usage of zfs-list faster when specific dataset info/properties are requested as some of them are bundled with the information queried at the initial stages of dataset lookup.

9/15/2020, 1pm Pacific

Meeting recording

- OpenZFS hackathon (Matt)
    - Spreadsheet with Hackathon Ideas/Breakout Sessions: https://docs.google.com/spreadsheets/d/1qH-qST3uSYVh7fzWHWlB1eM1y5rAYCQs6tRDjOhVYx0/edit#gid=0
    - Feel free to put your ideas in the doc even if you are just working by yourself!
    - Since it will be held virtually, the hackathon will be slightly more structured this year. Each idea proposer will be given a couple of minutes in the beginning of the day to

pitch their idea. Then a Zoom Room (and/or a Slack channel) will be created for each team project.

- Per-pool ARC STATS (Richard Elling)
    - Unlike the L2ARC, the ARC statistics are not broken down to per-pool statistics. It would be nice to be able to break them down into that fashion when dealing with issues.
    - For the short-term we could just add a few trace points but in the long term we may want something more integrated to ZFS.
    - This is still in the idea phase and the most straightforward solution so far seems to be adding more specific arcstats under the SPL kstat code (the upside of this approach being that there is no need for a userland change whatsoever).
    - There were also a couple of tangential issues brought up to the discussion:
        - The idea to potentially attempt to break this info even further to a per-dataset level
        - Existing aggsum implementation being slightly more heavy-weight and not as scalable as existing percpu counter implementations in FreeBSD and Linux. We could either learn from these platform-specific solutions and improve the aggsum code's design, or just get rid of its existing code and make it a thin wrapper on top of the existing platform-specific solutions.
- ZED all-syslog.sh (Don Brady)
    - Illumos' FMA logs were retained after a crash but Linux's zpool events are not. There exists all-syslog.sh in ZED but its output is terse and misses any useful information.
    - Don has prototyped a change in all-syslogs.sh to provide more useful information for some events (e.g. the vdev, and offset fields for checksum error events).
    - Even though changing that code may result in the breakage of some existing stuff, all agreed that this is a step into the right direction.
    - Don expects to open a PR for this soon.
- OpenZFS 2.0RC (Brian)
    - The first 2.0 release candidate was created 2 weeks ago and a new one is expected to open this week.
    - People who want to try it out are encouraged to do so while the code is soaking
    - There is no strict schedule for these RC releases but for now we expect to see one every two weeks approximately.
    - As an overall status, the project is still on track for the official 2.0 release before the end of the year.

8/18/2020, 1pm Pacific

Meeting recording

- OpenZFS Developer Summit (Matt)
    - Speakers have been announced and registration is open
    - First day will be talks, second day will be the hackathon
    - The hackathon this year will be slightly more organized, as participation is virtual. There will be a small idea pitching session at first before people break out to per-project sessions.

        - We are looking for people to lead the hackathon project sessions. If you are working towards something as part of the hackathon you can probably create a session for it and others can join to help.
- Boot once API changes (extend Delphix nextboot with nvlist support, want to get the signatures stable before 2.0) (Allan)
    - There is a working prototype which has undergone some preliminary testing
    - Allan and Toomas hope to have the API change in before the 2.0 release so it becomes part of stable.
- dRAID (Brian)
    - Distributed parity/spare implementation for ZFS.
    - The feature has been wrapping up - there is a PR that needs more reviewers. Users are also welcome to clone that branch and test the code.
    - Mark Maybee will give a talk about this.
- 2.0 branching (Brian)
    - Plan was originally to branch mid-August, we are close to that - the timeline has been stretched by a couple of weeks.
    - There is a possibility that DRAID will be part of it too.
- meta: move to or explicitly endorse semantic versioning for release versions · Issue #10334 (Gabriel Devenyi)
    - Discussion around having concrete information/communication for the patch/release branches of the code. E.g. What do patch releases contain? What branches do we maintain in parallel and for how long?
    - So far the policy has been that for major release branches we don't add new features or on-disk format changes, we provide fixes for kernel bugs and compatibility for new kernels, we sometimes include performance improvements if they are not very invasive. The cadence of those releases and their updates is roughly 3 months in general.
    - The point raised by Gabriel is that users may need some of these kernel/compatibility fixes earlier in some branches.
- L2ARC cache policy (Georgios)
    - Discussion started up on Github for the policy; MFU vs MRU, data vs metadata, etc.. - the main question raised was how do we move forward from here to allow users to get the most performance. Should we allow a kernel module parameter? A full-fledged knob from the userlard that is per-dataset? Should we just go ahead and change the default and leave it at that?
    - There are two guiding principles for answering the above:
        - [1] Is to incorporate more observability into the policy so we can make points through data-driven observations.
        - [2] If we are to provide a user-knob - we should be able to easily answer the question of what the knob does and when to turn it.
- dnode_sync is careless with range tree · Issue #10708 · openzfs/zfs (jclulow/pmooney; we would like someone to take a look!)
    - George and/or Matt will be taking a look at this

7/21/2020, 1pm Pacific

Meeting recording

- OpenZFS Developer Summit (Matt)

    - Event pushed 1 week later than the initial date - to the beginning of October
    - Talk Submission deadline extended by 2 weeks
        - If you have things you want to hear about feel free to ping Matt and he can bug the right people
- ZFS send compatibility for ZSTD (Allan)
    - Problem: Creating a stream that the receiving side can't understand (e.g. the receiving machine doesn't know about ZSTD because it uses an older ZFS version)
    - Multiple ways to go about it - there was common agreement that whatever the solution it should probably follow the following principles:
        - The receiving side should be able to tell the user whether it supports receiving the given stream, hopefully with a friendly error message
        - The default behavior in general should be to use the latest and most common thing but also give the option for the user to explicitly downrev/configure it to their needs.
            - E.g. people rarely do plain `zfs send` without -c
        - The above principles are not expected to be enforced/implemented for the ZSTD PR but they should guide the future direction of dealing with this issue. The ZSTD PR will proceed as discussed.
- Activating feature flags on 'zfs set' instead of first block birth (ZSTD) (Allan)
    - Problem: Activating a feature flag for a dataset, but not enabling it (e.g. haven't written any blocks yet), can cause panics when the dataset is opened by a ZFS version that doesn't know about that feature.
    - This issue has been brought up in the past, the next steps would be to have someone implement the fix.
- What are the plans for OpenZFS release? (Alexander Motin)
    - There are no features officially holding the release.
    - Brian will create the 2.0 version branch mid-August and probably make the release at some point before the end of the year.
    - The master branch will remain open for development during that time.
    - Any bug fixes applicable to the 2.0 branch will be "backported" from master.
- Forceful export - zpool export -F (Allan Jude)
    - Functionality: A pool's suspension doesn't cause ZFS operations to hang for other pools (e.g. no holding the namespace lock. etc..). In addition the suspended pool can be exported.

6/23/2020, 9am Pacific

Meeting recording

- OpenZFS Developer Summit 2020 (Matt)
    - Will be virtual due to COVID - dates 29-30th of September
    - Registration is open and FREE
    - Call for presentations is open
        - If you have done interesting work in/with ZFS we'd love to hear from you. Let us know by July 20th.
    - Talks will be recorded and will be streamed on the day of the presentation. Presenters will be in a live-stream at the same time to answer questions.

    - We'll try to emulate hallway/water-cooler discussions virtually. There will also be a virtual hackathon.
    - We're still looking into the technology to do this. If you have suggestions from other conferences please let us know
    - There are still sponsorship opportunities - 501c(3) tax-advantaged contributions
        - Even though the conference will be online doesn't mean that there are no costs
        - We are also trying to create different budgets such as paying for testing infrastructure or reduced ticket prices to future in-person conferences for underrepresented groups.
    - An email was recently sent on the mailing list with all the info
- block reference table for file cloning (Pawel)
    - File-cloning can be thought of as hard-links with copy-on-write properties
    - Use-cases include cloning VM images and other large files or being able to recover files from snapshots, fast and without additional space.
    - Implementation:
        - Dedup comes to mind as the end-functionality is somewhat similar. That said we would like this to be a generic feature of ZFS and not a feature flag. We'd also like to avoid the performance issues that come with dedup.
        - Current idea revolves around a new block-reference table, a mapping of vdev_id+offset to a refcount. The attempt would be to avoid overhead in reads/writes and just put the overhead in when freeing blocks with more than one reference (to make such operations performant we'd probably need to maintain most if not all of our structures/metadata in memory).
        - There are a few implementation questions that we'd need to iron-out. For example, do we expect this to work across datasets? If that's the case how would it affect send/receive assuming we want to maintain the cloning-property? What about memory? Can't that table grow too big? We'd probably need to think hard of how the table is managed and how to avoid unpredictable performance.
- Renaming nextboot feature to bootonce (Alan)
    - There is a naming conflict between the nextboot ioctl used by ZFS and the nextboot feature specific to the FreeBSD bootloader.
    - Proposal: Change nextboot ioctl name to bootonce
- Aliased system properties (Sean)
    - PR is out: https://github.com/openzfs/zfs/pull/10111
    - Still looking for reviewers and feedback
- Send toggle -L fix (Matt)
    - The original bug is discussed in the PR
        - Doing incremental sends where you mix enabling and disabling the -L flag can cause a bug that zeroes data.
    - The PR has been merged and now receives will do the right thing or fail to start.

5/26/2020, 1pm Pacific

Meeting recording

- OpenZFS conference
    - Will be held virtually/online this year

    - Currently planned at the end of September
        - there is flexibility as people won't be travelling
        - higher attendance is expected for the same reason
    - Talk Submissions is opening next week
    - Format will be roughly the same as other years
    - We still haven't decided on the platform to host it
        - We want to hear from others: what conference tools and formats worked on other conferences (or didn't) - contact Karyn or Matt
- Rebuild PR (Brian)
    - Work came out as preparatory work for the DRAID projects but for mirrors
    - The PR as it stands introduces a new option (-R) for the zpool replace and zpool attach commands that performs the rebuild and provides redundancy
    - The caveat is that no checksums are verified but for that same reason it can be a lot faster (especially for fragmented pools and/or small block sizes)
    - Each vdev can be rebuilt independently, so you can have multiple operations going on, unlike resilver which is pool-wide.
    - The code used to be part of resilver but then it was reworked to have its own standalone infrastructure.
    - Looking for feedback: https://github.com/openzfs/zfs/pull/10349
- Documentation system (George Melikov)
    - https://openzfs.github.io/openzfs-docs/
    - Short-term goal: Move Linux Wiki to the above repo and add automation
    - Long-term goal: Have all documentation at one place (e.g. man pages, content from the media wiki, etc..)
    - For folks interested in migrating the media wiki content feel free to ping Karyn, Michael Dexter, or Matt to see what content is still relevant and what is stale.
    - Open to feedback on what format and tools to use or other ideas
    - Several concerns with using RST as the source for manpages.
- POSIX AIO for async dmu? (Matt Macy)
    - "wondering if it's worth integrating async dmu with [POSIX AIO] as opposed to just using [async dmu] for zvol"
    - Some work was started by SpectraLogic years ago
    - The goal is to expose the asynchronous ZIO interfaces at the dmu level and integrate it to ZVOLs or the POSIX AIO interface (standard syscalls like POSIX reads cannot take advantage of this).
    - Regarding POSIX AIO the questions are:
        - Is anyone already using these interfaces? If so, what's their experience?
        - Would people be interested in using it if it existed? If so, for what use-case?
- Upstreaming "forced export" feature (Allan)
    - Work for this project will be starting soon
    - Use-Case: Problems in connectivity with remote vdevs would suspend the pool (and potentially block other pools when higher-level locks are grabbed)
    - Proposal: zpool export -F - force the export of that pool so everything goes back to normal.
- Formalizing pad2, bootonce/nextboot/rescue counter (Allan)

    - An interface introduced recently regarding this functionality only accepts a character array as an argument (in libZFS_Core). The proposal here is to either change this existing call or add a new one that accepts an nvlist instead to provide more flexibility for new features.
- OS-specific properties (Sean)
    - PR is out: https://github.com/openzfs/zfs/pull/10111
    - Deals with the issue of os-specific properties varying wildly in their behavior between platforms while there is room for normalization.
    - Needs Reviewers/Feedback

4/28/2020, 1pm Pacific

Meeting recording

- Libshare changes (George Wilson)
    - George has been doing work revamping the sharenfs property codepaths. The reason is performance and consistency between multiple consumers that need to look at the state of the system.
    - Work mostly combines what exists in FreeBSD while at the same time removing a lot of the logic around sharetab. The changes affect the FreeBSD code too where libzfs_fsshare is renamed/moved to libshare.
    - Preliminary Performance Testing (having ~1000 filesystems and multiple threads that share/unshare them) shows up to 99% performance improvements, on the scale of going from minutes to seconds.
    - During questions:
        - (Michael Dexter) Is iSCSI and/or FiberChannel on the radar?
            - There is some existing code to handle these subsystems but it hasn't been touched(used?) in a long time. It would be an interesting project if someone is using them and wants to pick up the work.
- Status on Panzura dedup (Josh P)
    - Initially thought to be self-contained but was discovered that this is not the case.
    - Internally within Panzura the effort has been temporarily deprioritized.
- Status on dedup-log & DDT limit (Allan)
    - Both coming along nicely. DDT is on the last round of feedback.
    - REMINDER: For the DDT limit the semantics have changed - it is not an explicit memory limit but more of an on-disk one. The memory-limit was causing usability problems and its behavior was hard to model.
- Persistent L2ARC merged
    - Feature doesn't require a feature flag.
    - There are still a couple of follow-up patches for that change that could use reviewers.
    - NOTE: The version merged is a slightly modified version of what initially had started from Nexenta.
- Dedup'd send/recv removed
    - NOTE: This feature has actually nothing to do with Dedup and is a specific feature of Send/Recv (zfs send --dedup/-D)
    - The feature has been removed from master. From 0.8.4 users should see a deprecation notice if they try to use it.
- OpenZFS for OSX common repo

    - Since FreeBSD was merged Jorgen is looking into a fresh new port for OSX to be merged into the OpenZFS repo. The new port gives us a chance to redo somethings that were not optimal in terms of implementation. The ability to migrate from old OSX pools to use the new implementation is not a requirement but it would be good to have.
    - Once everything is green with existing tests Jorgen will start pinging people for feedback, the potential of adding OSX tests in the CI pipeline, and to discuss issues found in the common code used by all the operating systems.
- Zpool user properties
    - Allan hopes to open a PR to upstream this soon.
    - Still looking to hear from people who already have use cases for this and overall any feedback.
- Changes that need reviewers:
    - Introduction of ZSTD compr. to ZFS - https://github.com/openzfs/zfs/pull/9735
        - Needs to be updated now that persistent L2ARC has been merged. There is also some leftover feedback to incorporate.
        - Allan is looking for feedback on how to break this change down to logical parts so it becomes easier to review, and potentially make it smaller.
        - There are also a few old ZSTD PRs laying around in the PR list of OpenZFS on Github, we should take one last look before closing them to ensure we haven't missed anything.
    - Dedup DDT load https://github.com/zfsonlinux/zfs/pull/9464/
        - Has been out for a while - we need to solicit the mailing list so we:
            - [1] Give a heads up that this is coming
            - [2] Get any high or low-level feedback
            - [3] Ensure that conflicts with other dedup patches that are still downstream are minimized.
    - Dedup Ceiling https://github.com/openzfs/zfs/pull/10169
        - Feedback from Matt is almost done and change should be ready to go soon.

3/31/2020, 9am Pacific

Meeting recording

- Add "zstream redup" utility; remove "zfs send --dedup" (Matt)
    - In ZoL version 0.8 a message was added to "zfs send --dedup" that the specific option will be deprecated
    - A userland utility "zstream redup" provides the exact same functionality now (taking a deduplicated stream and yield a normal one).
    - The "zstreamdump" utility is being deprecated too and its functionality is relocated under the same "zstream" command - "zstream dump"
    - PR is open and ready for feedback (https://github.com/openzfs/zfs/pull/10156)
    - The next part of this project would be to remove the deprecated functionality from the kernel.
- New API, higher-level than libzfs (Mike)
    - Goal: Expose the administrative interface of the CLI over an API - similar model to what AWS do with its CLI and API.

    - Motivation: Simplify integration with applications that want to administer ZFS.
    - Presentation Link: https://www.dropbox.com/s/gldjal30bw3he52/libzfs_api.pdf?dl=0
    - Takeaways:
        - People agree that this is a good idea, solves a real problem, and it is doable.
        - At first glance it seems like the majority of the work would be around error-handling as code from places like zfs_main.c would be moved to the library.
    - Next steps:
        - Get more folks involved as use-cases vary
        - Refine the proposal with more detailed and concrete examples
        - Share the above with the mailing list for more in-depth discussions
- Add O_DIRECT support: design update (Matt)
    - Updates since last meeting:
        - Incorporated feedback and made progress
            - One of the main questions from before: What are the semantics of concurrent stores to the buffer for the direct write? - the key thing is that the checksum should match the buffer, therefore if the user has checksums (and/or other related features enabled) a copy of the buffer is created.
            - More refinement of properties and misaligned operations
        - There is a design doc out discussing the different ideas and trade-offs: https://docs.google.com/document/d/1C8AgqoRodxutvYIodH39J8hRQ2xcP9poELU6Z6CXmzY/edit?usp=sharing
        - The doc has been shared with the mailing list for open discussion.

3/3/2020, 1pm Pacific

Meeting recording

- Zfs repo move (Matt)
    - Which repo should they use:
        - If FreeBSD / ZoL, you can continue to use the same URL. There should be link forwarding via GitHub, and you shouldn't have to do any local updates.
    - Code coverage links point to the wrong place, so they don't resolve. At least for PRs that were open at the time when the move happened. Probably this won't come up for new PRs.
- FreeBSD progress (Brian)
    - Over the past few weeks, they've been getting all of the tests running. That's mostly done.
    - There is one more big PR open to push the code (new files). Hopefully that will be merged this week or next.
    - Who is reviewing the code? Are reviewers needed?
        - FreeBSD folks have done reviews. Not waiting for any reviewers, but speak up if you have an issue.
    - Won't turn on CI by default until we know the tests are passing. You may have to manually initiate.
    - Illumos is watching the progress of FreeBSD closely and will then look at doing something similar with illumos.

- Directio (Mark M)
    - Motivation
        - Cray wanting to improve performance on NVMe drives
        - Sequential large-block read/write
        - Concerned with cost of bcopy to/from userland
        - With prototype, can saturate up to 12 NVMe drives (up from ~4 without directio)
    - implications of directly reading/writing
        - Less bcopy() of buffers
        - Less memory is allocated (e.g. no dirty data in dbuf cache)
        - Data is not added to ARC cache
        - Write happens before call returns (but data is not necessarily persistent if the system crashes)
        - Are directio reads truly zero-copy? How to compute checksum consistently? A: It isn't consistent. You could get a "false" checksum error (hardware is fine but software caused checksum to be wrong).
            - Maybe add a flag to the block that indicates that the checksum may be faulty because it was a directio write and userland may have modified the block while it was being written, causing a checksum error. Then handle checksum errors on these blocks differently.
    - User interface
        - Modes
            - Some mode that's the same as the current behavior (ignore DIRECTIO request, do it the old way)
            - Some mode that will be the new default, which does (at least some) DIRECTIO-requested operations directly.
            - (perhaps) some mode that does i/o directly even when DIRECTIO is not requested.
        - What happens to partial-block reads that are DIRECTIO-requested?
            - Fail?
            - Fall back to normal behavior, adding block to the ARC?
            - Do directly, discarding non-requested data?
                - performance impact to sequential partial-block reads
        - What happens to partial-block writes that are DIRECTIO-requested?
            - Fail?
            - Fall back to normal behavior, adding block to the ARC?
            - Fall back to normal behavior, but don't add block to the ARC?
                - I.e. write is buffered until sync context, potentially accumulating more changes, but after it's written in sync context, we discard it from the ARC/dbuf caches
            - Do directly (read-modify-write before returning)?
                - performance impact to sequential partial-block writes
                - (probably) implementation complexity
            - Nobody wants to argue against failing partial block writes
        - How does directio and regular i/o interact?
            - Inconsistent?
                - Matt says no
            - Consistent?

        - May be complicated to implement
    - fails?

2/4/2020, 1pm Pacific

Meeting recording

- Thread priorities on linux (Paul D)
    - Context: Performance analysis of ZFS send on ZFSonLinux found discrepancies with illumos
    - Root cause: Linux the threads are minimum priority, lower than user threads (as opposed to Illumos)
    - Workaround: Increased the priority for now
    - Question: How do we want to deal with this among different OS's?
        - FreeBSD has the same priority scheme as illumos (Linux being the outlier of the 3)
        - In Linux there were performance issues related with thread priorities in the past but there hasn't been any recent investigations
        - There is no silver bullet for this
    - Consensus:
        - There are not many thread cases whose performance we don't care about but we should at least break down threads into different groups and decide on the priority of each group
        - This issue is to be added to the list of issues that we need an open PR on GIthub for discussion
- Do we need to enhance the feature flag activation code. Setting checksum=sha512 or compress=zstd but not writing any blocks, can cause the pool to panic on older systems
    - Summary: Setting the compression to zstd but you don't write any blocks, and then re-import that pool in a ZFS version that doesn't understand zstd triggers an assertion failure
    - Consensus:
        - Bump the counter when the feature gets activated AND for each block created with it (refcount never goes to zero until the dataset is destroyed).
        - Although seemingly difficult to implement we should at least pay a close look at the code paths as it may not be as intractable as it seems.
- Support for ignoring (not being able to mount) datasets with unsupported feature flags
    - Consensus: This is a good idea in general but it should not be considered a solution for the aforementioned zstd problem.
- Moving of the OpenZFS website from Joyent to University of Washington is in-progress
    - No problems with Joyent, just planning for the long-term
    - No actual website changes, just the DNS endpoint will change
    - Still would be happy to update any contributions to update the website's content :)
- Changes that need reviewers:
    - Persistent L2ARC - https://github.com/zfsonlinux/zfs/pull/9582
    - Introduction of ZSTD compr. to ZFS - https://github.com/zfsonlinux/zfs/pull/9735
    - Performance Optimization for Encryption
        - Includes new algorithms and changing the default algorithm used
        - Action Item: Send an email to the mailing list as a heads up about changing the default before going ahead an applying that change

- FOSDEM, Scale, and other conferences
    - There doesn't seem to be any talks related to ZFS in the aforementioned conferences but people attending can submit BoF discussions
        - iXsystems will potentially be at Scale this year
    - Allan Jude has done a ZFS-related talk in the main track of FOSDEM in the past
    - There will be a ZFS BoF in this year's BSDCan and potentially one ZFS-related talk too.
    - The BSDNow podcast is open to interview people doing interesting work on ZFS
    - General Idea: It would be good to be active in conferences and groups outside of the OpenZFS summit and BSD
        - On this note, OpenZFS is considering spinning up funds for people that want to work on that (e.g. setting up a booth on those conferences and getting the word out).

1/7/2020, 9am Pacific

Meeting recording

- When will 0.8.3 ship and will it include the large-dnode "zfs diff" fix (FireSnake)
    - Currently there is an outstanding patch PR for 0.8.3 (#9776)
    - it should be landing when we wrap up our testing so very soon (probably within this month!)
    - The goal is to have it pulled in for Ubuntu and Debian LTS
    - Let us know if there is a patch that you really want to see there
- Changing the checksum algorithm at an existing dataset - even if no new blocks have been written - exporting and then importing that pool from a different version of ZFS hits an assertion error. [Allan Jude]
    - That was an oversight on the design of feature flags.
    - We should be able to handle this more gracefully.
    - A few ideas were thrown including:
        - Bump feature refcount for every dataset that has the property set
        - Being able to open the pool but not that dataset (property would show as "unsupported")
- Any further comments on this Ubuntu developer's proposal to enable encryption by default with a fixed passphrase: https://bugs.launchpad.net/bugs/1857398 (rlaager)
    - Is there any additional feedback that we haven't already provided them? Feel free to add in the thread.
    - Tangent on secure-erase and changing encryption keys:
        - Secure erase and not having dataset's being partially secure were two of the main motivators on the current encryption design.
        - The latter is ensured by setting encryption on at creation but not guaranteed as the encryption key may later be changed and will affect only new blocks.
        - Even if encryption key (user/wrapping key) is changed, new blocks can be read/manipulated if the old (compromised/public) key is known, and the old-key-wrapped master key is found (e.g. by forensic analysis of the disk).
        - The trade-off being usability and practicality varies wildly between cases.
    - Given the above tangent, we should really understand what the Canonical folks want to do and try to come up with the best practises and design. This includes

communicating best practices, and potentially implementing something different than what we currently have.
    - We need to be more clear about the security implications of “zfs change-key”
        - PR filed
- Change encryption=on from aes-256-ccm to aes-256-gcm? See especially the comments starting here: https://github.com/zfsonlinux/zfs/pull/9749#issuecomment-568633557 (rlaager)
    - The two main motivators of this proposal are security and performance.
        - From a security standpoint, Mozilla and TLS default to gcm.
        - According to Richard’s estimates, performance could get a ~3x improvement with gcm.
    - There seems to be an overall consensus on this but we should really check with Tom Caputi.
- encryption: from_ivset_guid check missing on resumed recv? (Christian Schwarz)
    - turned into issue post-meeting, do not include in upcoming agenda
- bookmark cloning & zcp bookmark support PR (Christian Schwarz)
    - open design question: what about redaction bookmarks?
    - reviewers needed once above question is resolved
    - outcome:
        - redaction object might be too large to copy it in syncing context, would have to be done in the background
        - => don’t implement redaction bookmark cloning at this point
        - => create thin bookmark: zfs bookmark #redactionbook #newbook
            - will not contain the redaction object itself

12/10/2019, 1pm Pacific

Meeting recording

- Saved send feature (Tom)
    - Be able to get a partially received dataset from one machine to another - do a ZFS “send of a partially received dataset”
    - Add-on functionality: ability to resume a saved receive from a bookmark
    - Status: Testing is almost done and Datto will soon start using it in production
    - Looking for extra reviewers for the PR - https://github.com/zfsonlinux/zfs/pull/9007
        - Paul Dagnelie and Matt Ahrens will take a look
    - Action Item: Send a heads up in the mailing list about the feature for other platforms
- ZSTD rework (WIP PR) (Kjeld)
    - The goal of this new PR is to get the best of all the existing forks of this feature
    - Status: Sebastien Gottschall (Brainslayer) have completed the majority of the design and thorough testing has been done. Code is being cleaned up and restructured.
    - Action Item: Sync with Allan who has also done some work with this
    - Action item: Get reviewers and feedback for the change
- Feature Request: Encryption to work with dedup across multiple datasets - Tom:
    - Today different “clone families” have different master keys (the key actually used to encrypt the blocks), so blocks with the same plaintext will have different cyphertext if they are in the same clone family - even if they have the same wrapping (user) key (i.e. same/inherited keysource property)
    - Want to add a mechanism to have the same master key for different clone families

    - Need to design user interface to make it clear what's going on.
    - Suggestion: use a property to indicate that all children have the same master key
        - Need to work out the details of how this would interact with things like rename (into / out of the "same master key hierarchy")
- Pull Request Open for Persistent L2ARC in ZoL - Brian Behlendorf:
    - This is a port of Sasso's/Nexenta's work (PR 9582)
    - Reviews requested
- ZoF update:
    - Getting closer! - Bulk of refactoring is done ~ approximately 4 files left to go

11/12/2019, 1pm Pacific

Meeting recording

- Rescheduled OpenZFS DevSummit talk: Securing the Cloud with ZFS Encryption (Jason King)
- New ZoL minimum kernel version (Brian)
    - Change: Increase the minimum version to 3.10 kernel (from 2.6)
    - Practical implication:
        i. Dropping some support for very old enterprise releases of Linux
        ii. Net removal of around 2,000 lines of code
    - Timeline: will not roll out until OpenZFS 2.0 version
- ZoL repo move/rename (Matt)
    - From ZFSonLinux to OpenZFS (mentioned at the conference during Matt's Talk)
    - Transfer ownership feature on Github is to be used (all PRs, issues, even URLs should be moved/redirected to the new home)
        i. If anyone has had a bad experience with this feature, please let us now
    - Should be a non-event for people doing development
    - Currently targeting to do this by the end of the year
    - Notes: We will keep all the existing ZoL resources like email list and IRC channel under the same name (ZFSonLinux, not OpenZFS) - OpenZFS mailing list will remain the canonical place for developer discussions for new features. Leave user-centered questions or platform-specific issues to per-platform mailing lists
- Issue tracker curation (Matt)
    - Request for input from folks: Several issues that have been filed against ZoL end up being open for a long time and contain a lot of off-topic discussion (some getting close to an actual flamewar). What should we do about this?
    - Consequences of this issue:
        i. Discourages new members from getting involved with the community
        ii. It is counter-productive for existing members- especially the ones who can fix the actual issue
    - Potential ways forward:
        i. Ignore these messages (status quo)

        ii. Try to respond to every message (re-iterate the current status, explain the situation or dependencies on other issues, etc..) - showing that we understand the situation - diffusing the situation a bit
        iii. Hide the off-topic comments - this could backfire a bit and cause more impolite messages.
        iv. A combination of 2 & 3 - where they do get a reply and then we hide the messages.
    - Idea for dealing with incoming bugs:
        i. After initial triaging or skimming through the issue, assign a specific tag or component and have a certain group of folks responsible for each component.
        ii. Have more folks help with labeling incoming issues
            - Unfortunately (and incredibly) github permissions only allow people with write access to the repo to do this.
    - Discussion
        i. The most popular "potential ways forward" were i (status quo) and ii (try to respond to every comment)
    - Action Item: Brian and Matt will draft a proposal for this and present it in the next meeting and the mailing list.

10/15/2019, 11am Pacific

Meeting recording

- Dev Summit
    - Please register if you haven't already - almost at capacity
    - Recorded and live-stream (same service as last year)
- zpool ddtload
    - A new subcommand proposed by Will Andrews
    - Use-case: Large DDT table - after reboot write-performance is bad because we have to read from the DDT
    - Proposed Solution: The new command will preload the DDT in the ARC on-demand
    - Current Status: Looking into plugging it in zpool wait so we know when it is finished and looking for reviewers. PR Link: https://github.com/zfsonlinux/zfs/pull/9464
- xattrs update:
    - Update from Gordon Ross:
        - Still gathering information. Currently looking for contacts and survey of how things happen today, especially for Windows and OSX (Jorgen Lundman is probably a good point of contact for that).
        - ACLs and DOS-style attribute bits are also an issue, but less critical. They are stored in a platform-agnostic way, but some platforms don't support them so this state is not visible on all platforms. Just a heads up, we are still trying to document those.

- Question: What is the status of OpenZFS repo - no sync with illumos a long time?
    - Currently planning to retire that repo - an email will be sent to let everyone know. The plan is to reinstate a repo with the same name (OpenZFS) but based on ZoL (and ZoF soon)
    - There is no real-deadline or schedule for this. Probably going to have it finalized once ZoF is part of ZoL and we have CI setup
    - More info on this and the status of ZoF at the Dev Summit
- Status of ZoF (ZFS on FreeBSD using ZoL codebase) (Post-meeting update):
    - Code is mostly written.
    - Working on upstreaming to ZoL. Probably at least a few more months to finish that.
    - Testing
        - ZTS runs on ZoF prototype
        - Still need to hook up buildbot to FreeBSD
        - Would like to upgrade buildbot
- Igork - Integration process idea:
    - Problem: The master branch has shown to have test regressions (or even panics) in the past. This is especially bad when you want to introduce a change and your PR fails for unrelated issues.
    - Idea: Have a separate branch (like a staging branch) and collect weekly/bi-weekly changes on it and test it thoroughly before merging it to 'master' (or potentially use different tag). Ideally we'd want a branch with no sporadic failures.
    - People agree that this definitely has been an issue in the past, but the proposed solution may increase the maintenance/coordination/release burden of our release/branching process. Additionally, testing may be incomplete as people test against a clean build but not the final product where everything will be merged together. This could surface new bugs and merging issues.
    - Another way would be to look into ways that we can tackle the same problem by having different kinds of testing, reverting commits faster, etc..
- Igork: It would be nice to see the drafts of the release notes of future versions. Helps people plan their upgrades accordingly for they things that matter to them.
    - We'd have to talk to the folks at Livermore who are currently organizing all the release logistics. Use of Milestones by Brian in Github achieves part of that but it is not a full solution to the problem. In general it would be nice to communicate more about branching and our release cycle.
    - Matt will work with Brian and others on this starting from the next release (transparent and communicate better, so we don't have to ping Brian personally)
- Possibility to move the "early" meeting even earlier (request from Ubuntu ZFS people in the UK who have family commitments at our usual times)
    - Current pattern: 2 at a later time for folks in Asia and 2 at an earlier time for folks in Europe. Proposal: 9am (2 hrs earlier)

- Most folks seem to agree that this is acceptable
- Plan: New meeting time to take effect from January onwards

9/17/2019, 1pm Pacific

- Meeting recording
- EOL ZoL on RHEL 6 (Brian Behlendorf)
  - RHEL 6 could be old enough that we could drop support for it on master (still supported for 0.8)
  - Technically will be EOL'd by Red Hat in November 2020.
  - Feedback from the community: Given enough Notifications beforehand, people should be fine
  - Actual change needed in ZoL:
    i. Go through build system code and remove any references of v3.10 kernel and older (new oldest supported kernel would be 3.11). The process should be similar on what ZoL did for deprecating RHEL5.
  - Action Items:
    i. Brian/Matt will give a heads up in the mailing list, in the release notes of each versions until then, open PR for this
    ii. We need volunteers for the build system changes
- Xattr cross-platform compatibility (Andrew Walker)
  - Problem:
    i. ixSystems works with services that receive alternate data streams written as xattrs in FreeBSD in the user namespace, which is implemented slightly different in Linux (there is "user." prefix - FreeBSD uses "freebsd." prefix(?) - Solaris uses "smb." prefix). Their application (Samba) is doing the same thing in Linux and FreeBSD, but ZFS represents them different on-disk between each platform. As a result, xattrs that are written in FreeBSD are visible in other OSes except from ZoL where the metadata disappears.
  - Potential Solutions:
    i. Brian: ZoL has around 4 prefixes, so one solution would be to have user as a fallback choice (e.g. if it is not part of any namespace, it is part of the user namespace).
    ii. Andrew Walker: Have a zfs dataset property to be able to tell which format is used
    iii. Andriy Gapon: Add some OS info on the actual attribute and have ZFS interpret them differently
      - Sef: Some form a feature flag that would fix the prefixes.
    iv. Matt Ahrens: First make it possible to read xattrs from all platforms, even if the names show up differently. A potential long-term solution: New stuff is written in some new format that is portable across platforms (e.g. in the zfs.* namespace) and each platform translates the ZFS prefixes to the local platform's prefixes.
  - Question: Is it an incompatibility between different OSes? or an incompatibility between different implementations of ZFS? Shall we have a translation layer outside of ZFS?

        i. A bit of both but mostly VFS layer (outside of ZFS code). Assuming it is only on the VFS layer, it would be reasonable to still have some way of accessing these attributes. A point for this, is that in ZoL there is little flexibility in changing the VFS code.
    - Action Items:
        i. Proposal & Next steps - Andrew can start a writeup and coordinate with Alexander from iXSystems
- Relax quota semantics for improved performance (Allan Jude)
    - Problem: As you approach quotas, ZFS performance degrades.
    - Proposal: Can we have a property like quota-policy=strict or loose, where we can optionally allow ZFS to run over the quota as long as performance is not decreased.
    - People's Feedback/Questions:
        i. Richard Elling - Isn't it the same problem when the pool is almost full (SLOP space)? Answer: This is slightly different, but the mechanism is the same, and we don't want to break that (e.g. run beyond SLOP space just like that).
    - Tangent: Should we scale the SLOP space appropriately? The SLOP space can bite a big chunk of space in big pools.
        i. Feedback: That seems reasonable, though the use cases may not be that many (fragmentation issues in such big pools will probably arise before encountering the SLOP space issue). See discussion on previous PR.
- zpool replace of a log (and maybe a cache) vdev – does this work well? Can it be improved? (Andriy Gapon)
    - Problem: a user had to replace a log device using the replace command and it took a long time (dozens of gigabytes were scanned). Can we do better? It seems like there is not special logic for devices like that, do we want to do something different for log vdevs? Even maybe prohibit using replace for these devices and advice the remove & add workflow.
    - Feedback: the above sound reasonable except for one thing. Log devices can have actual data on them. If you crash and you have blocks in the log device and you've removed the device, and you don't mount the specific filesystems, these blocks will stay there. Encryption should also make this more common. We need to retain the ability for the scrub-based replace/attach. We could improve the performance by looking at all the blocks of all the logs instead of looking at all the blocks in the pool.
    - Action Item: Andriy will look into this and create a doc
- Renaming bookmarks – are there any pitfalls? Seems like a useful feature that's not been implemented in a long time (Andriy Gapon)
    - Feedback: It should just work - one more thing to plumb through the CLI, libzfs, etc… internally, removing the ZAP entry and re-adding it with the new name should do the trick
- Panzura to open source their temporal dedup implementation (Josh P)
    - Panzura will be open-sourcing some parts of their self-contained ZFS implementation of temporal dedup on Github. There is hope from Panzura that this will be integrated within OpenZFS but at least for now there are no concrete plans of getting this code upstreamed without volunteers.
    - Question: What is temporal dedup?

i. A dedup scheme that groups blocks by the time that they are created/modified etc... Grouping blocks in such way should allow for faster access to the data due to caching based on temporal locality

8/20/2019, 1pm Pacific

Meeting recording

- OpenZFS DevSummit talks (matt)
  - Lots of great talks so far, but there is still room.
  - If you forgot to submit a talk before yesterday's deadline, please reach out right away. Matt will announce selected speakers in 2 weeks, so there is a small grace period.
  - Registration is open now, and Matt will publicize this more once we announce the speakers.
- Linux Plumbers conf (Serapheim)
  - George WIlson and Serapheim will be at the Conference if you want to talk ZFS or Debugging the Linux Kernel
  - If so, please reach out so the ZFS folks can meet up and talk about all things ZFS.
- OpenZFS on 32-bit architectures (relevant for FreeBSD, Linux, …): plain reads of 64-bit "atomics" (Andriy)
  - Issue discovered: We use atomic operations to modify 64-bit values that we treat as atomic, but to read them we use plain C assignment operations, which works ok for 64-bit platforms, but not for 32-bit. Those plain reads are implemented as 2 instructions there (32-bit), which is incorrect.
  - At this point it is mostly theoretical (its is rare to hit a write while crossing the 32-bit boundary) but it is still a valid concern.
  - The code paths are not very critical - mostly stats that we send to user space.
  - Proposal for discussion: Should we introduce some kind of format wrappers for those operations?
    i. Paul Dagnelie: Has done some similar work for the SPL back in the day and this should be straightforward.
       - Question: How many 32-bit platforms do we actually care for at this point?
    ii. Andriy: Have heard of a few 32-bit Intel users out there
    iii. Tom Caputi: Has some experience for compile-time checks for structures of these nature.
  - It seems like people are for it. Andriy will first talk with FreeBSD developers, figure out how they want to proceed, and try to propose designs common for all platforms.
- zfs share -a (George)
  - Issue: Stale file handles using zfs share -a and the sharenfs property, by restarting or rebooting the nfs server.
    i. The reason is that ZFS plugs into SystemD in a cumbersome/non-natural way.
    ii. NFS shares actually start before the ZFS service. Mountd responds to the client before the ZFS share is actually completed leading to the stale handles.

    - Hacky Approach:
        i. Introduce a new component to zfs share. Instead of zfs share -a , zfs share -g (short for generate)
        ii. The idea is to add ZFS logic that ties it closer to NFS
        iii. Change of dependency ordering, zfs share -g generates the handles for NFS, before NFS, and NFS consumes those generated handles.
        iv. Cons: A bit hacky for now
        v. Pros: Simple and fast
        vi. Plan: To do more redesign on top of this for a cleaner solution
        vii. Others that have similar issues with George/Delphix feel free to reach out.
            - Tom Caputi: Datto had issues with NFS in the past, and don't use sharenfs, but they are interested on having control over this functionality
            - Allan Jude: Seems like FreeBSD already does something similar, so there may be some convergence of goals.
    - Submitted this as lightning talk at the Dev Summit
- Anyone interested in working on implementing Matt's ideas for dedup ceiling and on disk format log (http://open-zfs.org/w/images/8/8d/ZFS_dedup.pdf) please contact allan@klarasystems.com to discuss. (Allan)
    - Problems using the feature for Terabytes of data due to known memory issues
    - There is budget for a contract to implement the limit of the size of the DDT proposed during one of the past Dev Summits
    - Feel free to reach out if you are interested in undertaking that work
    - Matt is willing to mentor whoever takes on this work

Attendees

- **Linux:** Brian Behlendorf, Tom Caputi
- **illumos:** Jason King, Jerry Jelinek, Joshua M. Clulow, Kody Kantor, Mike Gerdts, Roman Strashkin
- **Linux & illumos:** Don Brady, John Kennedy, Matt Ahrens, George Wilson, Paul Dagnelie, Serapheim Dimitropoulos
- **FreeBSD:** Alexander Motin, Allan Jude, Andriy Gapon, Josh Paetzel
- **OSX & Windows:**
- **Cross-platform:** Michael Dexter
- **Project management:** Karyn Ritter
- **Other attendees:** Rich Teer, Ryan Moeller

7/23/2019, 11am Pacific

Meeting recording

- ZoL + SIMD + Linux 5.0 (Matt)
    - Brian integrated some support for this already but he is currently on vacation
    - 5.0+ made some changes to the floating point registers that were used by ZoL for SIMD instructions
    - Brian reimplemented some of those in ZoL and now these instructions work again and will be backported in 0.8

    - Needed for performant RAIDZ, Encryption, and some compression algorithms
    - Used only if the underlying CPU architecture actually supported (checked during runtime)
- OpenZFS Developer Summit (Matt)
    - Announced: Nov 4-5. Same locations as last year in SF.
    - Registration and call for papers is open. Please submit a talk and tell us about the cool stuff you're doing with ZFS
    - We appreciate people volunteering to help out with things for the conference.
        i. Karyn will reach out to Rainbow about help with AV.
    - Ticket prices went up to $100. There are some discount tickets available, and let us know if the pricing is a hardship and we can try to figure something out.
        i. FreeBSD Foundation may have travel grants available if want to apply
    - Get all the details (including a link to the registration site) on the OpenZFS wiki page
- Property to track amount of queued TRIM/UNMAP (like "freeing" for async destroy) (Allan)
    - Allan Jude: users may want to know how many trim/unmapped is left or in-progress before performing other actions instead of performing the pool
    - Matt: Maybe a flag to zpool sync —trim that could wait for all trim activity to be done. Probably better than the existing proposal where we have a property like freeing for it, which would jump up and down
    - `zpool wait` that is about to be upstreamed from Delphix and should be a good candidate to add this functionality. You can look at the Delphix repo if you want to see it before the PR.
- RAIDZ removal issue with design ideas (Igor)
- RAIDZ expansion progress (Matt)
    - No progress to report since preview posted
    - The hope is to have takers to test it out
    - The functionality is there, there are on-disk format changes (so don't use it in production), a few race conditions being fixed and some more error handling for device failures.
    - Important to note that even though we can expand the RAIDZ we still can't remove devices (e.g. zpool remove)
    - ZoL PR exists with design considerations for the feature
- OpenZFS feature matrix (everyone)
    - FreeBSD is hoping to catch up with ZoL through the ZoF work
    - NFSv4 ACL PR waiting for code review - should we consider tracking this in the feature Matrix?
        i. Update: Added
        ii. Note: already on illumos

Attendees

- Linux: Andreas Dilger, Christopher Voltz, Garrett Fields, Tony Hutter
- illumos: Jason King, Jerry Jelinek, Joshua M. Clulow, Kody Kantor, Mike Gerdts, Mike Zeller, Sanjay Nadkarni
- Linux & illumos: Alek Pinchuk, Don Brady, John Kennedy, Matt Ahrens, Paul Dagnelie, Pavel Zakharov, Prakash Surya, Sara Hartse, Serapheim Dimitropoulos
- FreeBSD: Allan Jude, Andriy Gapon, Josh Paetzel, Kirk McKusick, Rainbow
- OSX & Windows:

- **Cross-platform**: Michael Dexter
- **Project management**: Karyn Ritter
- **Other attendees**: Chip Schweiss, Dipack Panjabi, Igor Kozhukhov, Marcin Skarbek, Neal Gomba, Ryan Moeller, sef, Steve Castano, zfslurker

6/25/2019, 1pm Pactific

Meeting recording

- FreeBSD / ZoL integration update
    - Goal: Single repo that builds for both platforms
    - https://github.com/zfsonfreebsd/ZoF/tree/projects/zfsbsd/module
    - Goal is for FreeBSD 13 to use ZoF by default (but may slip to FreeBSD 14)
    - Goal to get it into current repo around end of September?
    - Jorgen: Where should I direct Windows/OSX questions? (#openzfs or ZoL Developer list)
    - Do a PR for additional platforms/tree for additional platforms
    - Change zfsonlinux/zfs => openzfs/zfs. Brian & Matt will come up with a proposal for naming and structure that reflects what developers do/use
- Announcements of big, long-standing projects being available:
    - Linux - Redacted send/recv merged!
    - Illumos - ZFS crypto landed!
- Discuss error reporting (Tom)
    - Do we want to move to something more general (e.g., have the kernel return strings)?
    - Continue to extend zfs_errno_t, though this doesn't help provide additional context/information, and then add new strings for user
- pyzfs PR (Richard E)
    - Specifically how to get the pool config info programmatically
    - Programmatic "zpool status" that is retrievable, and any time you add something to zpool you need to make sure status continues to be retrievable. (some mistake that Joshua pointed out...)
    - Translation of the config to a stable nvlist? Not sure if this should be in libzfs_core or the kernel. Kernel is better but might ignore backward compatibility.
    - Newer version of userland continue to work without rebooting the kernel.
- Could a spanning mode option be added for multi-vdev zpools instead of the default striping data across all vdevs? The intent is to prioritize data integrity over access speed for large archival / backup pools, and a workaround with multiple pools joined with mergerfs seems suboptimal if the LOE of adding support for this is low. (Guirara DaiKaiju)
    - Matt: I think you're suggesting that `zpool create disk1 disk2 …` create the pool as a mirror, as though you typed `zpool create mirror disk1 disk 2 …`. And perhaps that `zpool add…` should instead do `zpool attach …`.

Attendees

- **Linux**: Andreas Dilger, Brian Behlendorf, Garrett Fields, Tom Caputi, Tony Hutter
- **illumos**: Jason King, Jerry Jelinek, Joshua M. Clulow, Joyce McIntosh, Kody Kantor, Mike Gerdts, Mike Zeller, RomanS, Sanjay Nadkarni

- **Linux & illumos**: Alek Pinchuk, Brad Lewis, Don Brady, Matt Ahrens, George Wilson, Prakash Surya
- **FreeBSD**: Alexander Motin, Allan Jude, Andriy Gapon, Josh Paetzel, sef (+Leo)
- **OSX & Windows**: Jörgen Lundman
- **Cross-platform**: Michael Dexter
- **Project management**: Karyn Ritter
- **Other attendees**: Christian Schwarz, Hajo Möller, Richard Elling JMoS

5/28/2019, 1pm Pacific

Meeting recording

- Happy ZoL 0.8!
    - Lots of downloads and excited people. Successful so far!
- Illumos encryption status (Jerry/Jason/Jörgen)
    - Jerry has all of Jörgen's work pulled over and building on illumos. Resolving some test failures and then will put it out for an illumos code review.
    - The code review will go out to the illumos developer mailing list and ask that they identify issues with the platform-dependent changes in the port. People will have ample time to comment and raise issues before the RTI is submitted.
- RAIDZ Expansion status (Matt)
    - Will have an Alpha out this week or next. Pools will need to be destroyed if you use it.
    - The change lets you add a device to an existing pool and reflows the data between disks.
    - All of the caveats and details of what does and doesn't work will be included in the PR.
    - Call for testing! Once the code is up, any manual testing under different workloads and setups will be greatly appreciated. It would also be great if people could help write systemic tests.
- BSDCAN update
    - The future of ZFS on FreeBSD (Allan)
        - One of the goals of the talk was to motivate the ZFS repo of FreeBSD rebasing over ZoL. The advantages of having the consumers of ZFS closer together was highlighted.
        - There was some initial pushback people thought that the rebase will make ZFS use the Linux Compatibility layer of FreeBSD, which is not the case. Once this was clear, there were no further pushbacks
        - Matt brought up that once the rebase is done and stable, discussions may be initiated for potentially renaming the ZoL repo.
    - snapshot scalability (Matt)
        - Slides are posted and video will be up eventually. Please check them out if you're interested in learning more about snapshots.
        - Matt will post a link to #openzfs once he has one.
- DRAID meeting recap (Richard Elling)
    - There were about 11 participants

    - There is a design doc (linked above) with all the history of its edits based on the discussions that took place. Please add comments to the doc to further the discussion.
    - There is still work to be done but since the code changes are sizeable, we'll try to split them into individual commits when it makes sense.
        - Sef says he is available to do reviews at any time, just reach out.
- CoC update (Matt/Karyn)
    - Code of Conduct was put into place since the last meeting. We made several changes based on community feedback. Thanks for participating!
    - We will have a call for nominees for the Working Group in the fall, and then will review the CoC at that point and have another community comment period. The goal is to complete this before the Dev Summit.
- The Dev Summit is Nov 4-5 in SF. See open-zfs.org for upcoming dates (e.g., call for presentations).
    - Registration opens July 8th
    - Send presentations to Matt by August 19th
    - Thanks to Datto, Delphix, and Intel for already sponsoring! Reach out to @kritter if you want to sponsor and she can give you the details.
- Request from sef! The On-Disk Format document is out of date, and doesn't reflect recent changes (e.g., encryption). It would be really helpful to update this and keep it up-to-date going forward (Perhaps in a semi-automated way?)
    - Tom: The info about how things look today exists in various places, so it is just a question of pulling it together into a single place.
    - Matt: We only have a pdf for that, and we probably can't just copy-paste it somewhere else. It would be good to have a document with this info. Let's put out a call for a volunteer to do this.
        - Paul pointed out that it is licensed under a Berkeley License
- Compressed arc performance issue - single threaded sequential read
    - Sequential reads are decompressed by the (one) user's thread.
    - With non-compressed ARC, they are decompressed by the (several) zio threads, leading to more parallelism and improved performance.
    - This is a good reason to disable compressed ARC (for workloads with single threaded sequential read).
    - We could make the prefetches decompress into the dbuf cache, to get this performance back.
        - If we do that, could we then make compressed ARC mandatory?
        - Maybe… but Allan may have found another way to address the problem that motivated him to make compressed ARC mandatory
- Please take a look at the redacted send-receive requests as soon as you get a chance so we can push this now that 0.8 is out. It's in sync with 0.8.
- Next meeting will be in 4 weeks: June 25

Attendees

- Linux: Brian Behlendorf, Tom Caputi
- illumos: Jason King, Jerry Jelinek, Kody Kantor, Mike Zeller, RomanS

- **Linux & illumos**: Matt Ahrens, George Wilson, John Kennedy, Paul Dagnelie, Pavel Zakharov, Prakash Surya, Sara Hartse, Serapheim Dimitropoulos
- **FreeBSD**: Alexander Motin, Allan Jude, Josh Paetzel
- **OSX & Windows**: Jörgen Lundman
- **Cross-platform**:
- **Project management**: Karyn Ritter
- **Other attendees**: jiyer, Kelly Hays, Richard Elling, Ryan Moeller, sef, Steve Castano

4/23/2019, 11am Pacific

Meeting recording

- DRAID meetup (Richard Elling)
    - This is an all-day workshop for people interested in using and developing the feature
    - The goal is to finalize the design and start working on it
    - May 3 at the Delphix SF office (343 Sansome St, Suite 900, San Francisco) at 9am
    - Please RSVP so we are sure to have enough space. Reach out to anyone on the team over email, Slack (#draid)
    - Karyn to set up Zoom
    - Richard will pull together an agenda document: remote folks (in particular) can add comments and items to the agenda
- ZoL 0.8 status (Brian B)
    - Tagged final release candidate last week. FC and soaking since then.
    - Please test it as much as possible!
    - There are a few remaining bugs they are trying to finish up, but there shouldn't be major changes.
    - Also working on documentation
    - Hoping to release in 2-3 weeks
- ZoL => FreeBSD: test images available (based on Dec ZoL)! See this announcement for details. Please test it out.
    - Allan will cross-post the ZoL on FreeBSD announcement to the openzfs list.
    - JoshP: The current ZoL-based implementation doesn't have TRIM enabled, are we dropping it?
        - Darth is waiting for ZoL to reach 0.8 and Delphix to upstream their latest features to ZoL before doing a final rebase for release. Hopefully in 4-6 weeks.
            - This rebase should also include the TRIM changes.
- Any feedback on Porting ZoL commits to other platforms? (Brian B)

  Are there things we can do to make this easier? Problems found during porting?

    - Igor: Identify pre-platform features, functions, etc. where things have diverged between platforms. How to report issues in a way that provides visibility to all of the platforms?
        - BB: They've tried to keep things consistent with illumos, but have likely diverged. Getting some documentation from illumos and other folks would help. Source code? Wiki? Mailing list?

    - Jerry J: Has been documenting divergence in the SPL interfaces (that he noticed as he was porting from ZoL to illumos) in the and can send that out to the mailing list and we can figure out a better place to put it.
- Enable compression by default at pool creation (Issue) (kpande)

  The idea is to give a better out of the box experience. The downsides are: potential CPU cost (and write performance due to compressing) and new users misunderstanding how the space is used in certain cases.

    - Igor: There have been issues when root pools are expecting compression to be disabled (on SPARC?)
    - Allan: This has been the default in FreeBSD installer for ~2 years
    - Sef: Boot pools need to be different for each platform, but grub doesn't seem like it needs to be different. Sef supports it being the default.
    - Compression=on has been the default in FreeNAS for several years; no complaints
    - *** Someone to take an action item to investigate benefits and issues, and write it up? (Matt will put out a call for someone to take ownership of this)
- Next meeting: push back 1 week (to May 28)?
    - Done!
- Code of Conduct
    - Send feedback by Apr 29!
- Allan: Proposal to upgrade the LZ4?
    - In general people are in favor of this
    - The first step is to just update decompression so people get the performance improvements
    - Next is the exploration of enabling the compressor potentially with a tunable that can fall back to the old decompressor
        - The fear is that enabling the new compressor could increase the space usage if using nopwrite (e.g. double it for existing snapshots)
    - Igor: How stable is LZ4+compression?
    - Allan: No known issues that he is aware of.

Attendees

- Linux: Brian Behlendorf, Garrett Fields, Tom Caputi
- illumos: Joshua M. Clulow, Mike Gerdts, Kody Kantor, Jason King, Jerry Jelinek, Patrick Mooney, Sanjay Nadkarni, Mike Zeller
- Linux & illumos: Matt Ahrens, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, Don Brady, George Wilson, Pavel Zakharov, John Kennedy, Paul Dagnelie
- FreeBSD: Alexander Motin, Josh Paetzel, Allan Jude, Darth
- OSX & Windows:
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees:, zfslurker, sef, Richard Elling, Rich Teer, Christian Schwarz, Steve Castano, Igor Kozhukhov, Ryan Zezeski

3/26/2019, 1pm Pacific

Meeting recording

Next meeting will be at 11am PT

- Log spacemap is out for review (Serapheim)
    - Open PR for this work, including performance results
    - For people who want to port this feature to another platform, Serapheim also has a list of other changes required
- DRAID summit (Richard Elling)
    - This work has been progressing, and it is at the point where it would be great to get people together in a conference room (physically and virtually) for a day to hash it all out. We would do this in the May/June timeframe.
    - Please raise your hand (click yes in the participants window) if you want to participate.
    - Working on finding the venue.
    - Yes votes: Joyent, Mark Maybee, BrianB, Don Brady, Matt Ahrens, Tom Caputti. Please reach out to Richard if you are interested in participating.
- Default pool features (Josh P)
    - Summary of the proposal
        - Portable, current, and what was available in 20XX + Tier 1 platforms
        - Tier 1 platforms: FreeBSD (11.X), ZoL (0.7.X), illumos-gate (from 1 year ago)
        - zpool status or upgrade would use the previously selected portability setting
    - Questions raised:
        - Ability to define the feature sets at runtime, so you can add new definitions on your own
            - Sef: Disagrees with the editable file. My own suggested addition would be the ability to query an existing pool against portability list. e.g., "How portable is this pool?" I don't know if that's useful outside my head.
        - Tier 1 platform: specific ZoL version rather than a specific OS platform
        - Root pools: grub doesn't support the same features as the Tier 1 platforms.
        - Matt's take:
            - User-configurable features may or may not be useful, but we can hold off and decide on that later.
            - ZoL: Which distros are shipping their own versions and how important are they?
            - Grub: We discussed this before, and we decided it was too hard and we were going to defer the conversation until later
        - Do we want to bias toward portability or the latest feature set?
        - Each distro ships different versions of ZoL:
            - RHEL has a separate repo for ZoL (0.7)
            - Debian ships 0.6.x
                - Matt: If you aren't shipping your own version of ZoL, you aren't a Tier 1 platform?
        - Since portability isn't turned on by default, maybe it's fine

        - You could say that only RHEL & SLES (+Ubuntu) are the platforms that matter since they are large, supported releases. Focus on main distros and release versions
        - Ubuntu LTS will be Tier 1
        - Christian: Heard a rumor that Canonical is working on an installer that will include ZoL. Is that happening? What will that mean for portability?
            - Canonical are working on it, but we don't have all the details.
        - Jorgen: Considering that "upgrading" is one-way - you need to be conservative. At the moment, users want to dual boot Windows and Linux, and don't care they are missing "userobj quota" as it's not exactly going to make the pool faster.
        - JoshP: Boot pool is not nearly as important as data pools
    - No volunteers to lead the implementation effort.
        - Suggestion to have the discussion on the mailing list to make sure people are included.
        - Matt pointed out that the person leading the implementation gets to decide the details beyond what's in the core requirements. (subject to code review)
- Feature idea: GUIDs for filesystems that are invariant to zfs send | recv (Christian)
    - `guid` is publicly documented since https://github.com/zfsonlinux/zfs/pull/6102
    - The `guid` property for snapshots is invariant to zfs send | recv
    - I use `guid` in zrepl to build diffs of the snapshot lists between "the same" filesystem on the sending & receiving side.
    - However, filesystem identity is currently derived from the dataset path. The `guid` for a sent-recvd filesystem is different on the receiving side, hence not invariant.
    - Filesystem identity across pools / machines is required to reliably track renames & destroys of filesystems for purposes of replication.
    - Question: Do we want such a cross-pool `guid` for filesystems _in_ ZFS?
        - Previous work & ideas in that area?
        - How does a receiving pool handle an (unlikely) collision of GUIDs?
            - Do we have this problem for snapshots already?
        - Alternative: implement it as a user-property.
    - JoshC: How does -R work with this?
        - Matt: It is in userland, and it does work. When you send, it has the from guid and you send it to the new guid. You don't necessarily know the name if it has been renamed, but you can find the snapshot that has the same guid as the from guid that the send stream is describing. You can receive the same snapshot in one pool. The problem is solvable with the solution today.
        - Christian: Exhaustive search is good, but it can be ambiguous if we want to properly distinguish divergence from deletion & recreation (=> proper error messages, etc). Wants reliable detection mechanism.
    - No clear conclusion on this, and Christian also doesn't have time to implement this on his own at this point.
        - Matt is open to someone implementing a guid for zrepl / other ZFS-specific functions.
            - For pjd's need: Exclude these sub-filesystems or metadata. Use redacted send/receive (+ more stuff)

Attendees

- Linux: Brian Behlendorf, 2 LLNL people, Tony Hutter, Carles Mateo, Garrett Fields
- illumos: Joshua M. Clulow, Kody Kantor, Jason King, Jerry Jelinek, Joyce McIntosh, RomanS
- Linux & illumos: Matt Ahrens, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, Don Brady, George Wilson, Pavel Zakharov
- FreeBSD: Pawel Dawidek, Kirk McKusick, Alexander Motin, Josh Paetzel
- OSX & Windows: Jörgen Lundman
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees:, zfslurker, sef (+cat), Richard Elling, Rich Teer, Christian Schwarz, Jason, Ryan Moeller, Steve Castano, Mark Maybee, Dan Langille

2/26/2019, 1pm Pacific

Meeting recording available here

- Reviewers for fast clone deletion (ZoL PR; illumos PR) (Sara)
    - There is a feature flag change that Sara sent email out about.
    - Sara is seeking reviewers.
        - Brian volunteered to review, initial pass looks great
    - How much review is needed?
    - Will have some conflicts with bpobj iteration work.
    - BB: Wasn't planning to get this in before 0.8, but if it would be useful it is possible.
        - Let's get it in right after 0.8 (which is due out in March)
        - There are a few fixes that are pending for 0.8 to go out. Matt will look at them!
- FIPS 140-2 certification (Luke)
    - Defense contractors could use ZFS for many things, but require FIPS. Other industries (e.g., healthcare) also have this requirement.
        - JC: Can you get certification for a source or is it a specific binary build?
        - Rainbow: It is for specific binary builds, and she does a lot of compliance and can help here. You can do source code level certification.
        - sef: It is really expensive and time consuming. Level of configuration for testing and certification is super specific.
        - BB: We do have binaries from companies like RHEL, but they aren't official builds."FIPS verified" rather than certified? We're already using the appropriate crypto algorithm.
        - PD: Certifying at the source level makes it easier for a vendor to get certified. There are some additional components that would probably need to be looked at (like hash algorithms).
        - MA: Certification applies to the crypto algorithm. Does that help us since it is a separate module.
        - AJ: Different Linux distros will have different binaries that would need to be certified separately.
    - Luke can connect with Rainbow and sef to see what would need to be done to see if it is viable.

- Should compressed ARC be mandatory? (Issue) (Allan J)
    - Does anyone turn off compressed ARC? If not, we can avoid special cases for this.
    - Please let Allan know right away if you do turn off compressed ARC. Else that functionality may just be taken out.
        - MA: Seems like there are some cases on Linux where we can be confident that people aren't using this combination (i.e., it doesn't work), but that doesn't cover all cases.
        - AJ: The crypto changes definitely made it different than what was in FreeBSD.
        - JC: FWIW, compressed ARC makes ARC better in many different ways in Postgres (at least). They haven't noticed any latency increases or memory overhead that has been called out.
        - AM: It is pretty pointless to disable compressed ARC. The difference when you disable it and use other mechanisms (e.g., bcopy), is negligible and there are other benefits to keep it on.
    - Please comment in github!
    - There was no significant negative feedback, so we plan to move forward with making compressed ARC mandatory.
- Platform-specific sharenfs (George)
    - Sent this out the proposal last night.
    - Create platform-specific properties. These platform-specific properties won't take effect when importing the pool on a different platform.
        - MA: 'sharenfs' is a system property because ZFS takes action based on it (e.g. share/unshare when you do 'zfs rename'). It should be platform-specific because the value of the property isn't verified/interpreted by ZFS - it's passed to the OS-specific share utility without modification.
        - AJ: Cross-OS import is a feature I'd like to keep as a 1st class citizen.
        - MA: JC provided feedback on the proposal. Please talk about your counter-proposal.
        - JC: Biggest difference was to keep the functionality the same if people really want it, but there would be a "veneer" interface that would allow platform-specific properties. Generally people would just use whatever properties on their OS. Garrett's comments are an appliance-centric view. The key bit of the proposal is to ensure that it isn't dangerous to import onto a different platform. The better eventual solution is to actually do something good in this scenario.
        - AJ: What about namespacing the property like we do for the user properties: sharenfs:illumos. Maybe make it more feature flag. Do it in a way that is consistent with what is already done.
        - CS: bikeshed: org.openzfs.illumos:sharenfs
        - RE: There could be many different iSCSI servers, and would need to figure out how to get to the right one. Some sharing is cross-OS: samba, nfs-ganesha
        - AJ: share@samba, share@nfs-ganesha, share@illumos… -- Tie this to the server rather than OS?
        - JC/GW: Maybe a "." instead of "@"?
        - sef: Who is going to decide if this is an OS-specific property?

        - CS: Have some hooks into a layer (e.g., via lua) rather than having this built into zfs.
    - MA: Seems like George's proposal is better than what we have now, but we should get feedback from various platform vendors.
- Next meeting will be at this time. The following meeting will be at the earlier time.

Attendees

- Linux: Brian Behlendorf
- illumos: Joshua M. Clulow, Kody Kantor, Jason King, Jerry Jelinek, Joyce McIntosh
- Linux & illumos: Matt Ahrens, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, Don Brady, John Kennedy, George Wilson, Pavel Zakharov
- FreeBSD: Rainbow, Pawel Dawidek, Allan Jude, Kirk McKusick, Alexander Motin
- OSX & Windows: Jörgen Lundman
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees:, zfslurker, sef, Richard Elling, Rich Teer, Jeff, Christian Schwarz, Luke Olson, evanl, Garrett Fields

1/29/2019, 11am Pacific

Meeting recording available here

- Code review progress on redacted send/recv (Paul D)
    - Still need people to review chunks to minimize code churn
    - BrianB: What is the schedule for 0.8? This wouldn't make .8 because they are trying to get this out in Feb/March (as quickly as possible).
- TRIM update (Brian)
    - User visible change is to align CLI with vdev … Everything else is under the covers
    - Looking for reviewers this week! Please review it if you have time.
- Update on default pool features to enable (Josh P)
    - Bootpool hasn't been considered. Proposal to include thoughts about this to spark the broader discussion.
    - Josh P will send email with a consensus proposal. Please send feedback / questions. As needed, discuss at the next meeting
- Remove dedup-ditto (Matt)
    - This is very little used and not very useful. The option still exists, and the system can deal with existing pools. But users won't be able to use this functionality going forward.
    - Silence and encouragement for removing this.
- Remove zpool remap (Matt/Brian)
    - It is possible to get value out of this in certain circumstances, but there are lots of caveats. Also, the cost to maintain it is significant since there is a bug that causes a panic if you're removing a device (remap) and a receive hits at the wrong time.
    - Seems like this functionality also isn't much use, and isn't helpful. This has already been removed from Linux since there were no tagged releases (and it was new to Linux). Command line options were removed in Linux, but the code is there and disabled. At some point in the not-too-distant future, the code would be removed.

    - Propose removing it from illumos & FreeBSD (shipped in 12.0 in Nov/Dec).
    - Josh C: Seems fine to remove the mechanism. For illumos, they will probably want to think more about what to do with the command (deprecation warning, etc)
    - Allan Jude: Seems to make sense to neuter it as soon as possible so that people don't start to rely on it. Similar to Linux, there should be and exit 0 and error.
- Log spacemap (Serapheim)
    - Upstreaming of this feature, which improves random writes. It is an on-disk format change
    - Opened multiple PRs in ZoL to break down the changes.
    - Aiming to have a pull request for the feature end of this week / beginning of next week.
    - There are a lot of metaslab changes.
    - Igor: What about upstreaming to OZFS and illumos?
        - Serapheim: Planning to do this work after ZoL, and will do it as a separate change. Mostly the code is the same. Will give an update at the next meeting.
- Fast clone deletion (Sara)
    - Originally developed on illumos as well. Pull request for OpenZFS and ZoL.
    - Performance enhancement for doing clone deletion, and tying it back to the number of changes on the clone.
    - PR has a lot of information about the design and implementation.
    - Igor: Please share performance details about this change.
        - Sara: Perf results are in the PR, will add test code for reproducing the results
- How many event/fault management frameworks exist? Can they be unified? (Michael Dexter)
    - On illumos is system-wide, not just ZFS. Not sure about other platforms.
    - Igor: Should we port FMA to other platforms?
    - Don B: In Linux, they emulated enough of FMA to get things working, and there is a README in that code that provides some details. In Linux, they had ZFS Event Daemon (ZED), and that allowed them to share the same modules as illumos.
    - Josh C: FMA provides details about faults and their resolution. It goes through sysevent to a daemon. Events are collected across the system, and then you can correlate errors from across the system and ties those to issues in the vdev layer. It would likely be problematic to put all of this into ZFS because the broader context of the system and source of the error would be missing.
    - Josh P: FreeBSD also has some emulation. There was a lot of effort put into getting things working, and he would prefer not to touch it at this point.
- OpenZFS/openzfs github repo (Matt)
    - Primary motivator to create this repo: Make it easier to get changes into illumos for non-illumos developers (e.g., porting ZFS changes from one platform to another).
    - Secondary motivatory: Gives visibility to the project and community rather than having changes and activity distributed across platform-specific repositories.
    - The primary motivator is still valid and valuable. It is also a lot of work to maintain it.
    - Delphix is spending the most time on this, but we're going to be focusing on Linux going forward.
    - Do people find it useful? Are others willing to take this on / help?

        - Josh C: Joyent is looking to set up and maintain infrastructure this year (especially around testing), and provide facilities and drive engagement. You're welcome to rename it to "illumos"-something.
        - Allan J: Does it make sense to also use it to help coordinate changes across platforms? Like changes to on-disk format.
- Mixing raw/non-raw send streams (Tom)
    - Last(?) encryption issue (that we know of!)
    - Tom is working on the PR

Attendees

- Linux: Brian Behlendorf, Andreas Dilger
- illumos: Joshua M. Clulow, Kody Kantor, Sriram Narayanan, Jerry Jelinek, Sanjay Nadkarni, Joyce McIntosh, Roman
- Linux & illumos: Matt Ahrens, Paul Dagnelie, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, Don Brady, John Kennedy, George Wilson, Alek Pinchuk
- FreeBSD: Rainbow, Josh Paetzel, Pawel Dawidek, Allan Jude, Kirk McKusick, Alexander Motin
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees:, zfslurker, sef, Igor Kozhukhov, Dan Langille, Richard Elling, ram, Rich Teer, Ryan Moeller, Tony Hutter, wombat

1/8/2019, 1pm Pacific

Meeting recording available here

- sharenfs property:
    - Currently wrong, because this is an illumos-specific string that it is up to other platforms to interpret it.
    - For now, this property is implementation dependent.
    - Linux folks barely use it, mostly for simple scenarios. In FreeBSD the effects of this property varies between system versions.
    - **Conclusion/Next Steps: George Wilson will make a proposal and send it to the mailing list.**
- Persistent L2ARC:
    - No activity on this for the past few years.
    - Currently, no high priority for anyone involved in ZFS.
    - Conclusion: We leave the status as it is, until some new need/effort comes up.
- TRIM Support:
    - Currently two implementations. One already merged in FreeBSD and another one in PRs to illumos and Linux originating from Nexenta.
    - FreeBSD users have seen some performance improvements. Same for some Nexenta folks.
    - The common pattern being that this is very vendor specific and having TRIM on can boost or degrade performance.
    - The complexity that the TRIM change introduces is fairly significant at the same time, so hard evidence is needed to convince going through with those PRs.

    - Conclusion: We can continue with this once we have some hard data on its behavior and how it helps from existing users of TRIM.
- Anyone using LSI HBAs with many SSDs or NVMes on FreeBSD contact allan@klarasystems.com

12/4/2018, 1pm Pacific

Meeting recording available here

- Default pool features to enable. A couple of options:
    - All features are enabled
        - Tom: This is the right thing to do because otherwise people miss key features.
    - Compatibility flag for platform and version
        - Sef: This is a horrible, non-scalable solution to actually list everything. Instead, have a tag which the platform can map to a specific release/set of features.
        - Joshua Clulow: Thought sef's idea was more manageable.
        - Neal Gompa: Downstream from ZoL users (e.g., Ubuntu) could see some issues because kernel and userspace tools may be out of sync and cause troubles.
    - **No** features on by default
    - All, none, least-common denominator ("portable pool")
- Moving the next meeting to Jan 8 (rather than Jan 1), but not changing the time.

Attendees

- Linux: Brian Behlendorf, Tom Caputi, Garrett Fields, Andreas Dilger, Neal Gompa
- illumos: Joshua M. Clulow, Mike Gerdts, Kody Kantor, Jason King, Patrick Mooney, Mike Zeller, Sriram Narayanan
- Linux & illumos: Matt Ahrens, Paul Dagnelie, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, Pavel Zakharov, Don Brady, John Kennedy
- FreeBSD: Rainbow, Andriy Gapon, Josh Paetzel
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees: Joyce McIntosh, zfslurker, sef (+ lovely black cat), Chris Johnson, Igor Kozhukhov, Chip, Dan Langille, Glenn Lockwood, Galaxy S8, Jerry, Richard Elling

11/6, 1pm Pacific

Meeting recording available here

- Status update on encryption ports
    - Update from Jason King: continuing to try to reproduce reported issue
    - Update from Sean Fagan & Matt Macy: still seeing some ztest failures, need to investigate
- Status update on redacted send/recv code review
    - Requested reviews from Tom, Roman, Brian
    - Tom requests breaking down review into different parts

- Discuss the OpenZFS Feature Matrix (by platform): details in this spreadsheet
    - Attempt to assign owners for all features “integrated to at least one platform”
- Openzfs github emails - vote against reinstating the notifications
- Meeting logistics - vote to extend future meetings to 1 hour
- New ZFS on Linux committer - it’s Matt Ahrens!

Completed Action Items

Karyn to change calendar invite to 1 hour meetings going forward, and add as many attendees to the calendar invite as possible.

Status: (11/6) Done. Attendees of today’s meeting with a * next to their name have not been added to the calendar invitation because I don’t have their email address.

Attendees

- Linux: Brian Behlendorf, Tom Caputi, Tony Hutter(?), *Garrett Fields
- illumos: Joshua M. Clulow, Mike Gerdts, Kody Kantor, *Jason King, *Patrick Mooney, *Mike Zeller
- Linux & illumos: Matt Ahrens, Paul Dagnelie, Serapheim Dimitropoulos, Sara Hartse, Prakash Surya, George Wilson, Pavel Zakharov
- FreeBSD: Allan Jude, Alexander Motin, Rainbow, *Kirill Ponomarev
- OSX & Windows: Jörgen Lundman
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees: Sanjay Nadkarni, *Joyce McIntosh, *zfslurker, *tech, *sef, *Roman, *Ryan Moeller, *Tim Chase, *Darth, *Chris Johnson, *Andreas Dilger, *Neal Gomba, *Rich Teer

10/9/2018, 1pm Pacific - Meeting Kickoff

Agenda

- Discuss the OpenZFS Feature Matrix (by platform): details in this spreadsheet
    - Attempt to assign owners for all features “integrated to at least one platform”

Notes

Meeting Recording is now available

Michael Dexter, for iXsystems:

My lab now has six identical Xeon E3 systems with two disks each for testing of different platforms in parallel: Windows, NetBSD, FreeBSD 12, Encryption on FreeBSD, Centos and someone suggested OpenIndiana as a go-to Illumos.

Updated the spreadsheet with feedback. Main updates were to:

- Zpool initialize on linux
- Encryption on illumos
- MMP on illumos
- Sequential scrub/resilver on illumos

We received feedback that 30 minutes was not enough time. That’s true but I’m hoping that we didn’t miss anything urgent, and we’ll get to the whole spreadsheet in the next few months.

Attendees

- Linux: Tom Caputi, Carles Mateo, Tony Hutter
- illumos: Bryan Cantrill, Joshua M. Clulow, Jerry Jelinek, Kody Kantor, Mike Zeller
- Linux & illumos: Matt Ahrens, Don Brady, Paul Dagnelie, Serapheim Dimitropoulos, John Kennedy, George Wilson
- FreeBSD: Andriy Gapon, Allan Jude, Warner Losh, Kirk McKusick , Alexander Motin, Josh Paetzel, Rainbow, Kirill Ponomarev
- OSX & Windows: Jörgen Lundman
- Cross-platform: Michael Dexter
- Other attendees: Chris, Richard Elling, Glenn Lockwood
- Project management: Karyn Ritter

Team Members

- Matt Ahrens - Linux & illumos
- Brian Behlendorf - Linux
- Don Brady - Linux
- Bryan Cantrill - illumos
- Tom Caputi - Linux
- Joshua M. Clulow - illumos
- Paul Dagnelie - Linux & illumos
- Michael Dexter - Cross-platform
- Serapheim Dimitropoulos - Linux & illumos
- Richard Elling -
- Andriy Gapon - FreeBSD
- Tony Hutter - Linux
- Jerry Jelinek - illumos
- Allan Jude - FreeBSD
- Kody Kantor - illumos
- John Kennedy - Linux & illumos
- Glenn Lockwood -
- Warner Losh - FreeBSD
- Jörgen Lundman - OSX & Windows
- Carles Mateo - Linux
- Kirk McKusick - FreeBSD
- Alexander Motin - FreeBSD
- Josh Paetzel - FreeBSD
- Rainbow - FreeBSD (Moogsoft)
- Karyn Ritter - project management
- George Wilson - Linux & illumos
- Pavel Zakharov - Linux & illumos
- Mike Zeller - illumos
- Chris -

Full Attendee list (baseline)

- Linux: Andreas Dilger, Brian Behlendorf, Carles Mateo, Christopher Voltz, Garrett Fields, Tom Caputi, Tony Hutter
- illumos: Jason King, Jerry Jelinek, Joshua M. Clulow, Joyce McIntosh, Kody Kantor, Mike Gerdts, Mike Zeller, Patrick Mooney, RomanS, Sanjay Nadkarni, Sriram Narayanan
- Linux & illumos: Alek Pinchuk, Don Brady, John Kennedy, Matt Ahrens, George Wilson, Paul Dagnelie, Pavel Zakharov, Prakash Surya, Sara Hartse, Serapheim Dimitropoulos
- FreeBSD: Alexander Motin, Allan Jude, Andriy Gapon, Darth, Josh Paetzel, Kirill Ponomarev, Kirk McKusick, Pawel Dawidek, Rainbow, Warner Losh
- OSX & Windows: Jörgen Lundman
- Cross-platform: Michael Dexter
- Project management: Karyn Ritter
- Other attendees: Chip Schweiss, Chris Johnson, Christian Schwarz, Dan Langille, Dipack Panjabi, evanl, Glenn Lockwood, Igor Kozhukhov, Jeff, jiyer, Kelly Hays, Luke Olson, Marcin Skarbek, Mark Maybee, Neal Gomba, ram, Rich Teer, Richard Elling, Ryan Moeller, Ryan Zezeski, sef, Steve Castano, Tim Chase, zfslurker