klara

The Future of OpenZFS and FreeBSD

EuroBSDCon 2019, Lillehammer, Norway
allanjude@freebsd.org

klara

Summary & Introductions

1

Allan Jude

FreeBSD Core Team
OpenZFS Developer

2

Klara Inc.

FreeBSD Professional
Services and Support

Covered in this presentation

OpenZFS Past, Present, Future

Challenges Facing OpenZFS

Changing FreeBSD's Upstream

What does the future hold for OpenZFS?

The Evolution of ZFS

- ZFS was originally developed at Sun Microsystems starting in 2001, and open sourced under the CDDL license in 2005
- Oracle bought Sun in 2010, and close sourced further work
- illumos, a fork of the last open source version of Solaris became the new upstream for work on ZFS
- ZFS was ported to many platforms, including FreeBSD in 2007 and Linux in 2008. The OpenZFS project was founded to coordinate development across platforms.

OpenZFS

- The original plan for OpenZFS was a single common repository where the OS independent code would live and be tested
- Each OS would pull in this repo and maintain local glue bits
- However, the effort required to maintain a repo that would not be directly used by any of the consumers was too great
- The “repo of record” became a copy of the illumos repo, pull requests were carried through illumos’ RTI by Matt’s team
- FreeBSD tracked this repo very closely, commit by commit

Platforms

- OpenZFS is now available on almost every platform
  - illumos (OmniOS, OpenIndiana, SmartOS, DilOS, Tribblix)
    - FreeBSD (FreeNAS, XigmaNAS, TrueOS, pfSense, etc)
      - NetBSD
    - ZFS-on-Linux (Ubuntu, Gentoo, Proxmox, OviOS)
      - Mac OS X (https://openzfsonosx.org/)
      - Windows (https://openzfsonwindows.org/)
      - OSv

Divergence

- Each different platform's version of ZFS started to diverge
- OpenZFS replaced the old "pool version number" with "Feature Flags", since features would land in different orders
- Bugs were fixed in one repo and not necessarily upstreamed or communicated to other platform's could apply the same fix
- Each camp did their development within their own community, and other communities might not be aware of duplicate efforts, etc.

OpenZFS Developers Summit

- The new OpenZFS project organized a conference in November 2013 to have developers from the various platforms share their work and future ideas and find solutions
- Included a platform panel (Linux, Mac OS, IllumOS, FreeBSD) and vendor lightning talks
- Attended by over 30 developers, since grown to over 100
- Now includes a hackathon to work on prototypes while experts are in the room for advice / design discussions

Leadership Meeting

- At the OpenZFS Developer Summit 2018 a discussion between the various platform leaders lead to the formation of a monthly video conference to discuss ongoing issues
- Meeting online once a month, and in person yearly, provides more information exchange and faster response times
- Goal is to keep the platforms better in-sync and compatible, and to get input from all platforms during design phase.
- Open to anyone. Live streamed and recorded to Youtube

Outcomes

- The leadership meetings have been very successful
- OpenZFS is working to standardize the command line interface where it has diverged across platforms
- New features are discussed during the design phase and platform specific issues are resolved early, with better results
- More effort into effective naming of tunables (ashift is an internal implementation detail, the user tunable should be called sectorsize and be expressed in bytes)

Does This Mean The Same To You

- NFS options are different on every OS
- How to make them more common so pools can move between platforms and have the same functionality
- Extended Attributes are different on each OS (and even each FS on each OS). A pool created on FreeBSD and then imported on Linux will have its xattrs seem to disappear
- Solving this could be fixed in the OS on FreeBSD and illumos, but ZoL and ZFS-on-OSX have no control over the OS

Deprecation Policy

- Creation of a deprecation policy. After 18 years it is time to remove a feature from ZFS: Deduplicated send (replication)
- Feature is not related to pool dedup, rarely used, complicated
- This will require building a utility to convert old replication streams so they can be received by future versions of ZFS
- Also slated for removal: dedupditto. Designed to write a 2nd copy of a block if it is deduped more than 100 times. Turns out it has never worked properly, not checked/fixed on scrub

Platform Deprecation

- ZFS-on-Linux is planning to drop support for RHEL/CentOS 6 in the next release (0.9), including removing the support code from the master branch
- If openzfs remains available as a port on FreeBSD we will need to define a policy for what is supported as well
- Currently the port only works with 12.1 and later due to requiring support for new algorithms in OpenCrypto

Compression Conundrum

- I am working on ZStandard compression of ZFS
- For LZ4, ZFS bundles a specific (old) version
- ZStandard is under very active development, we do not want to be frozen to an older version
- Need to support upgrading compression code
- No guarantee data will compress to same hash

Cross Platform Compatibility

- Improved user interface for pool creation. Specify `compat=openzfs-2019` or `compat=freebsd-12`
- Enable only feature flags compatible with that platform
- The OpenZFS-YYYY macro will refer to what is available across all platforms as of January of that year
- Still identifying what best options are for other values
- Need to suppress upgrade nag messsage in `zpool status`

Upstream Change

- Over time, bulk of new development has shifted away from illumos and towards Linux. This resulted in most new features arriving in the ZFS-on-Linux repository first
- Rather than wait for these changes to be ported to illumos before they can be made available on FreeBSD, it was decided to shift our upstream to the ZoL repo, so new features would be available on FreeBSD sooner

The Big Lift

- Once we decided to switch upstream, the question was how?
- Trying to incrementally reduce the differences was going to be very laborious and error prone. ZoL has been adding local changes and feature development all through the process of catching up to modern OpenZFS.
- This makes porting individual features from ZoL to FreeBSD very difficult. Even for relatively small features (MMP)

Re-porting

- There are so many changes and additional minor features that it was too difficult to untangle them all.
- Instead, it was decided the best path forward was to port ZoL to FreeBSD. This would provide all of the new features contained in ZoL, and a minimal diff so that future changes can be pulled in much like they are today from illumos
- A snapshot of ZoL from December, ported to FreeBSD, went out for testing in April. Have you tried it yet?

Fighting the FUD

- This news resulted in some immediate negative gut reactions
- There is only one OpenZFS, we are all in this together
- FreeBSD will get features sooner, be more involved upstream
- Linux uses the SPL (Solaris Porting Layer) to run ZFS code as close to the illumos upstream as possible
- FreeBSD does similar, so using ZoL code will not inject Linux or GPL code into the FreeBSD kernel
- No Linux-KPI shims are used for ZFS code

OpenZFS Realized

- This moves us to, and even beyond, the original goal of a single common repo, having 'OS Dependent' and 'OS Independent' code, similar to the MI/MD split in kernel code.
- The upstream repo will contain OpenZFS, and directories for linux and freebsd, and in the future the other platforms. We will leverage the CI work that has already been done, and this will mean that all proposed changes will have to pass CI on Linux and FreeBSD before they are merged.

FreeBSD Specific Bits

- TRIM: For years FreeBSD was the only platform with TRIM for ZFS. However upstream now has a superior one based on work out of Nexenta. It has better queuing and supports both online (constant, like the FreeBSD one) and on-demand TRIM.
- Jails: Based on ZFS Zones support, retained in os/FreeBSD
- NFSv4 ACLs: Retained as OS Dependant code

Features: All Platforms

- sequential scrub/resilver
- zpool scrub pause/resume
- device removal
- zpool checkpoint
- zpool initialize
- spacemap encoding v2
- Channel programs
- large dnode

Features: Some Platforms

- Encryption (incl. raw send/recv)
- multi-import protection (MMP)
- special devices for metadata (allocation classes)
- parallel ZFS mount
- zpool sync
- TRIM (new way)
- resilver restart
- xattr=sa

Features: Coming Soon

- fast clone deletion
- spacemap log
- remove dedupditto
- redacted send/recv
- ZSTD compression
- per-vdev properties
- Enable compression by default

Features: Future

- RAID-Z Expansion
- DRAID
- Persistent L2ARC
- Temporal Dedup (thanks jpaetzel@ and Panzura)
- Adaptive Compression (compress more when not busy)
- Smart Compression (file based heuristic)

Features: Wish List

- Improved dedup (dedup log)
- Offline Dedup
- File Cloning
- Per-dataset throttling/QoS (IOPS and BPS)
- SMR Support (Shingled Disks)
- Platform specific ShareNFS property handling
- Clustered Features
- Continuous Replication

Get Involved

- The OpenZFS community is very active and very welcoming
- Watch some of the past “OpenZFS Leadership Meeting” conference calls on youtube to see for yourself
- The “repo of record” is transitioning to the ZFS-on-Linux repo
- Github Issues and Pull requests
- Mailing Lists (Topic Box) for discussions
- Developer Slack channel
- Join the monthly call

klara

QUESTIONS

klarasystems.com

More Resources

- Want to know more about ZFS?
  - “FreeBSD Mastery: ZFS” & “FreeBSD Mastery: Advanced ZFS”
  - Not just for FreeBSD, DRM-Free ebooks ZFSBook.com
  - https://www.FreeBSD.org/handbook/zfs.html
- BSDNow.tv - Weekly video podcast on BSD & ZFS
- @allanjude on twitter