ZFS Draid Workshop Agenda
3-May-2019

The OpenZFS community has been experimenting with a new declustered RAID (draid) vdev type. The primary focus has been on rapidly rebuilding failed HDDs to ensure maximum data protection in the event of disk failure.

Logistics

Venue: Delphix San Francisco office, 343 Sansome St, Suite 900, San Francisco, CA
Remote access via zoom: https://delphix.zoom.us/j/918968072 or by phone (US)+1-669-900-6833
Time: 09:00 - 16:00 US/Pacific

For non-Delphix employees, please RSVP to Karyn.Ritter@Delphix.com, mahrens@Delphix.com and Richard.Elling@RichardElling.com in order to get pre-approved for access.

Attendees

Brian Behlendorf
Mark Maybee
Richard Elling
Matt Ahrens
Alek Pinchuk
Tom Caputi
Kais
Stuart Maybee (remote)
Jeff Yana (remote)
Don Brady (remote)
Carles Mateo (remote)

Agenda

- Welcome and Introductions
- Where we are and how we got here: current status and a little bit of history
    - the labs contracted with Intel to do draid work
    - Isaac presented at OpenZFS Dev Summit in 2015, later updated in 2017

    - at one time there is a sequential rebuild by metaslab for mirrored pools
- Observations and notes from practitioners
- Current code available
    - Isaac's PR -> Don's PR
    - Don's PR is the most up to date, based on ~0.8rc1, Oct 2018
    - Richard has been internally testing based on Isaac's PR
    -
- Goals
    - what is the MVP?
        - Create draid with (only) zpool command (no draidcfg)
        - clean up zpool status output
        - no draid-mirror, use special vdevs instead
        - config nvlist in label (need to double check if this is already done)
            - To fit the config in the label, perhaps limit the number of permutations, rather than limiting the number of devices
        - target workloads: large I/O, streaming
        - reasonable defaults to prevent harmful configs
        - zed logic for logical spares
    - what would the perfect draid look like?
        - small block workload solution (beyond using a separate special vdev)
        - draid-mirror
        - Sub-draid
        - Make compression code aware of minimum allocation size, and not do compression if we can't save at least one allocation (which may be quite large) - zio_write_compress() needs to take into account DRAID min alloc size, not just ashift
        - global physical spare intelligence
- Gap analysis
    - The three dRAID repos
        - Isaac's original branch - simplistic view
        - beta combined with allocation class
        - rc1 combined with another allocation class

          The Intel CORAL dRAID development repositories are now on github.
          https://github.com/don-brady/coral-beta-combined
          https://github.com/don-brady/coral-rc1-combined

          An example commit (most recent):
          https://github.com/don-brady/coral-rc1-combined/commit/4492717277c8ab99c0b7ad52627a84d105810b5d

    - need to reconcile all of the branches

        - compiles properly on redhat
        - need to capture bug fixes
        - todo: compare the branches
    - format is git archive, need to get into a repo form for review
    - Current code does not do sequential rebuild if there is not a virtual spare available
        - Want to make this do sequential rebuild with a normal (physical) device replacement as well
- Deep dives
    - draid pool life cycle
    - zpool command
    - Spares
    - zdb
    - ztest and zloop
    - ZFS test suite
    - FMA, zed, and events
    - Performance
    - Debug messages
    - Documentation and best practices

Deep Dives

draid pool life cycle

1. Plan
    a. Logical spares
    b. Recordsize and volblocksize
        i. Propose to default recordsize to 1MB if draid vdev is present. Logic here.
        ii. Space efficiency spreadsheet
        iii. verify zvols are excluded from special class
    c. Special class for metadata and small blocks
    d. draid mirrors
    e. copies (ditto blocks)
    f. compression: is the current 12.5% goal appropriate?
2. Create
    a. Today: draidcfg
        i. verbosity
        ii. base permutations (combinations)
    b. zpool create (details below)
3. Replace disks
    a. Logical spares
    b. Physical disk replacement

4. Expand
    a. Expand disks (is this practical?)
5. Add top-level vdevs
    a. Add 2nd top-level draid
    b. Mix with other vdev types
6. Remove vdevs

zpool command

I think this is the most recent consensus:
A new way of specifying the draid geometry, which can specify either `D` (number of data disks) or `G` (number of groups):

`draidPp+Dd:Ss`, e.g. `draid2p+8d:3s`

or

`draidPp:Gg:Ss`, e.g. `draid2p:3g:3s`

If `D` is specified, then the number of specified disks must divide evenly, otherwise we'll give an error. If `G` is specified, then the number of data disks will be computed based on the number of disks provided, and the number of data disks per group may differ. In this case, any number of disks (at least `(1+P)*G+S`) may be provided. When displaying the draid vdev configuration back to the user (e.g. in `zpool status`), we'll print it the same way the user specified it. (However, the initial implementation may not support specifying `G`, depending on how much time we have.)

To show the resulting config, we want some verbose mode that will show one line per group (and one line for spares) to show how many disks are in each. E.g.: "`zpool list -v --draid-groups`" could show:

```
draid2p:3g:3s
  Group 1: 8 data + 2 parity disks
  Group 2: 8 data + 2 parity disks
  Group 3: 7 data + 2 parity disks
  Spares: 3 spare disks
```

Proposal (not consensus yet):

```
  pool: testpool
 state: ONLINE
  scan: resilvered 3.02M in 0 days 00:00:04 with 0 errors on Fri May  3 10:43:26 2019
Draid groups:
Draid2p:3g:3s-0:
  Group 1: 8 data + 2 parity disks
  Group 2: 8 data + 2 parity disks
  Group 3: 7 data + 2 parity disks
  Spares: 3 spare disks
```

```
config:

       NAME            STATE     READ WRITE CKSUM
       testpool        ONLINE       0     0     0
         draid2p:3g:3s      ONLINE       0     0     0

           sdd19       ONLINE       0     0     0
           sdb2        ONLINE       0     0     0
…
           sdd18       ONLINE       0     0     0
       spares
         %draid3-0-s0  AVAIL

errors: No known data errors
```

todo: Remove draid cfg= option

todo: adjust "draid3-0" to reflect actual config

todo: zpool list -v to give more description of draid config (`zpool list -v --draid-groups`)

Q: what is the limit to config size due to label space A: we're ok

Q: can we expose a minimum allocatable block size for the top-level properties?

todo: warn if zvol creation with blocksize too small. Perhaps increase the default volblocksize to at least the minimum allocation size?

Spares

1. Logical spares
    a. Is there a use case for no logical spares?
        i. Yes, if the rebuild can work for physical replacements
        ii. currently "zpool replace" is the UI for rebuilds and resilvers
    b. Should logical spares be required? No
2. Physical spares
    a. Given logical spares, is there a need for physical spares?

```
root@visd:~# zpool status
  pool: testpool
 state: ONLINE
  scan: none requested
config:

       NAME            STATE     READ WRITE CKSUM
```

```
        testpool        ONLINE       0     0     0
          draid2p:8d:1s-0      ONLINE       0     0     0
            sdc1        ONLINE       0     0     0
...
            sdc11       ONLINE       0     0     0
          draid2p:8d:1s-1      ONLINE       0     0     0
            sdb1        ONLINE       0     0     0
...
            sdb11       ONLINE       0     0     0
        spares
          %draid2p:8d:1s-0-s0  AVAIL
          %draid2p:8d:1s-1-s0  AVAIL
```

The door is open for future UI experts to improve with later PRs

zdb

1. long options (separate from draid)
    a. only propose long options for draid options
2. revisit metaslab dump options for draid rounding
3. elide the permutation table unless multiple -l (ells)
    a. describe as a 2-dimensional array
4. Q: is there a better way to describe space utilization?
5. label audit for all leaf vdevs (separate from draid)

ztest and zloop

1. zloop changes
   Added dRAID specific section to zloop to exercise dRAID across a wide range of configuration parameters. For example:

   ztest -VVVVV -K draid -D 7 -G 6 -S 1 -m 0 -r 1 -R 1 -v 0 -a 12 -T 84 -P 38 -s 384m
    - ztest updates in master are desired
    - ztest time limits might need to be adjusted
    - todo: verify scale up for /var/tmp to test larger draid configs
    - todo: determine if ztest is different between draid branches

2. ztest changes
   New command options to control ztest testing with dRAID:
   -K draid|raidz|random -- kind of RAID to test
   -D <value> -- dRAID data drives per redundancy group
   -G <value> -- dRAID redundancy group count
   -S <value> -- dRAID distributed spare drives
   -R <value> -- RAID parity (raidz or dRAID)

ZFS test suite

1. changes to common shlib
    a. state addition: rebuild, distinct from resilver
    b. resilver aka rebalance
    c. creating many partitions
    d. other helpers
2. proposed draid.run
    a. long running tests
    b. repetitive replacements
    c. fill and test on large pools
3. add draid pools into existing tests (eg. autospare, setup default draid)
4. do random configurations work?
5. add negative tests
6. add test for restarting rebuild or resilvering

FMA, zed, and events

1. Sparing policy
    a. todo: verify logic for sparing
2. Checksum diagnosis
    a. todo: verify errors are assigned to the correct disk
    b. Q: should we zinject to existing tests?
    c. verify existing corruption repair works

Debug messages

1. draid_dbg and printk: converted to zfs_dbgmsg
    a. get newlines right
2. eliminate extraneous dbg in hot paths
3. debug levels
4. DTRACE_PROBEs and bpftrace (not MVP)
5. kstats (not MVP)

Performance

1. ZTS performance tests
    a. none currently exist
    b. none proposed, due to complexity of analysis
    c. agree that performance regressions are not desirable
2. Does draid make sense for all-SSD pools? Yes

Documentation and best practices

1. MVP
    a. zpool man page
    b. zdb man page
    c. ztest man page
    d. vectorized parity?
2. Long term
    a. when to use
    b. how to use
    c. understanding space allocation

Logistics

AI: Richard Elling will create a branch of his personal repo, folks can do PR's against that

Who has cycles to work on this?

- Tom / Datto has time to work on sequential rebuild
- Brian / LLNL will have time in a month or 2
- Mark / Cray has sporadic cycles
- Richard has "some cycles"

Brian can own rebasing the repo and coordinating code reviews, starting in a month or 2.

Break big mongo merge into smaller pieces

- sequential rebuild
- draid feature
    - ZTS tests
    - ztest/zloop
    - man pages
    - zpool
    - zed
- Extensions beyond MVP (e.g. draid-mirror)
- zdb long options
- ZTS common shlib

Dangling Participles

Q: is vectorized calculations correct?
Q: gang block handling (not MVP)?
Q: vdev ops xlate: take read code and adjust to fit
Q: understand rebuild throttle: do we need to assign to scrub class?

References

https://github.com/zfsonlinux/zfs/wiki/dRAID-HOWTO
draid installation and configure doc

Pull requests to zfsonlinux repo:
https://github.com/zfsonlinux/zfs/pull/7078 (from Isaac)
https://github.com/zfsonlinux/zfs/pull/8016 (updated version from Don)
https://github.com/zfsonlinux/zfs/issues/3497

Don Brady's October 2018 merge of draid code to zfsonlinux master, branch=draid_rebase
https://github.com/don-brady/zfs/tree/draid_rebase

Richard Elling has a version of 8016 rebased onto master as of March 2019

Presentations about draid

DRAID by Isaac Huang
https://www.youtube.com/watch?v=xPU3rIHyCTs

DRAID Performance by Charles Mateo
https://www.youtube.com/watch?v=CVST0t0X9YU