

OpenZFS on illumos

Where OpenZFS Originated

- 2001 – Started at Sun
- 2005 – Released through OpenSolaris
- 2010 – illumos spawned; fork of OpenSolaris
- 2013 – OpenZFS created
- OpenZFS's "home" is in illumos:
 - Due to history, but also OS integration: grub, mdd, fma, etc
- But OpenZFS is growing beyond illumos

Development model on illumos

- Committer access is granted to "advocates"
- Advocates rely on "reviewers" to verify changes
 - E.g. for correctness, good design, etc
- No explicit releases
 - All changes must be "release quality"
- Development tools and processes are difficult
 - E.g. patch, compile, deploy, test is cumbersome

How to facilitate collaboration?

- We encourage "upstreaming" changes
 - Difficult with current development model
- How can we make collaboration easier?
 - We're open to changes in development model
 - Peer code reviews are good
 - High overhead to build and test is bad
- Would an OpenZFS repository help?
 - If so, what are the requirements?
 - How can we get there?

Perspective coming from ZFS on Linux

- Large overhead for ZFS on illumos changes
 - ZFS on illumos is tightly integrated with illumos
 - illumos is the kernel, libraries, and more
 - Overhead for "lone" developer is prohibitive
 - ZFS on Linux is isolated, little dependencies
- Full illumos build: ~2 hours
 - Build ZFS only: ~6 minutes
- ZFS on Linux build time: ~3 minutes

ZFS on Linux to illumos (continued)

- Kernel tools are generally much better
 - mdb is awesome! crash probably could be.
 - pipelines and walkers
 - "SQL for crash dumps"
 - dcmds allow extensibility
 - ZFS specific extentions
 - `::walkers, ::findleaks, ::stacks -m zfs, ::whatls, ::spa, ::dbufs, ::blkptr, ::zio_state`
 - No gdb; no line number resolution
 - kmdb and dtrace are also very helpful

ZFS on Linux to illumos (continued)

- Smaller community of ZFS users on illumos
 - People involved are more informed
 - Fewer number of people testing
- ZFS test suite available on illumos
 - But, no xfstests or filebench

mdb Example – ::spa -v

```
> ::spa -v ! head -n 15
```

```
ADDR          STATE NAME  
ffffffff096151a000  ACTIVE rpool
```

ADDR	STATE	AUX	DESCRIPTION
ffffffff095050c780	HEALTHY	-	root
ffffffff09505106c0	HEALTHY	-	/dev/dsk/c2t0d0s0
ffffffff09630ac000	ACTIVE	tank	
ffffffff096be74540	HEALTHY	-	root
ffffffff09616f34c0	HEALTHY	-	/dev/dsk/c3t0d0s0
ffffffff09629c9780	HEALTHY	-	/dev/dsk/c3t1d0s0
ffffffff096be6f900	HEALTHY	-	/dev/dsk/c3t2d0s0
ffffffff096be6f280	HEALTHY	-	/dev/dsk/c3t3d0s0
ffffffff096be6ec00	HEALTHY	-	/dev/dsk/c3t4d0s0
ffffffff096be6e580	HEALTHY	-	/dev/dsk/c3t5d0s0

mdb Example – ::spa -Mh

```
> ::spa -Mh ! head -n 15
```

```
ADDR          STATE NAME  
ffffffff096151a000  ACTIVE rpool
```

```
ADDR          STATE    AUX          DESCRIPTION  
ffffffff095050c780 HEALTHY  -          root  
ffffffff09505106c0 HEALTHY  -          /dev/dsk/c2t0d0s0
```

```
ADDR          FRAGMENTATION  
ffffffff095986b740          32%
```

```
9:    113 *****  
10:   131 *****  
11:   391 *****  
12:   456 *****  
13:   250 *****  
14:   227 *****  
15:   386 *****
```

mdb Example – ::dbufs

```
> ::dbufs ! wc -l
182819
> ::dbufs | ::print dmu_buf_impl_t ! head -n 15
{
    db = {
        db_object = 0x76
        db_offset = 0x1a4a0000
        db_size = 0x20000
        db_data = 0xffffffff03b2dcd000
    }
    db_objset = 0xffffffff0991377c00
    db_dnode_handle = 0xffffffff09e0266d58
    db_parent = 0xffffffff09e4b22808
    db_hash_next = 0
    db_blkid = 0xd25
    db_blkptr = 0xffffffff09e21a5280
    db_level = 0
    db_mtx = {
```

mdb Example – ::dbuf

```
> ::dbufs | ::dbuf ! head -n 15
      addr object lvl blkid holds os
fffffff0af2001010      76 0      d25 0 tank/fish
fffffff0c26001018      84 0      68c 0 tank/fish
fffffff0c260010f8      77 0      1e9 0 tank/fish
fffffff0af20011d0      71 0      dc5 0 tank/fish
fffffff0c260011d8      65 0      b55 0 tank/fish
fffffff0af20012b0      7e 0      fb8 0 tank/fish
fffffff0c260012b8      80 0      a8a 0 tank/fish
fffffff0c26001398      b7 0      a2b 0 tank/fish
fffffff0af2001470      6e 0      91e 0 tank/fish
fffffff0c26001478      86 0      834 0 tank/fish
fffffff0af2001550      85 0      e05 0 tank/fish
fffffff0c26001558      87 0      851 0 tank/fish
fffffff0af2001630      6a 0      353 0 tank/fish
fffffff0c26001638      74 0      49d 0 tank/fish
```

mdb Example – ::whatis

```
> ffffffff09e4b22808::what is ! head -n 15
ffffffff09e4b22808 is allocated from dmu_buf_impl_t:
      ADDR      BUFADDR      TIMESTAMP      THREAD
      CACHE      LASTLOG      CONTENTS
ffffffff09e50de9c0 ffffffff09e4b22808      3ed28b4787 ffffffff09beccc840
                  ffffffff095b1c0448 ffffffff090f2b6900      0
                  kmem_cache_alloc_debug+0x2e0
                  kmem_cache_alloc+0x2d0
                  dbuf_create+0x5a
                  dbuf_hold_impl+0x177
                  dbuf_findbp+0x17b
                  dbuf_hold_impl+0xf9
                  dbuf_hold_level+0x31
                  dbuf_hold+0x21
                  dmu_buf_hold_array_by_dnode+0x109
                  dmu_read_uio_dnode+0x5a
```

End