Porting Ceph to FreeBSD: A look at designing code for portability

Willem Jan Withagen
Digiware

Contents

- Personal introduction
  - Why does life leads to Ceph
- Where do I want to go
  - FreeBSD, ZFS, bhyve
- Porting
  - Trivial stuff
  - Simple problems
  - Hard stuff
- Things to do
- Questions

Personal stuff

- Elektrical Engineer from TU Eindhoven
  - Worked as a system architect at Philips Research
    - Used Apollo Domain, HPUX, VAX, Sys3, and what not more
- Started 2nd ISP in the Netherlands in 1993
  - Ran in on FreeBSD 1.0
  - Sold it in 2000, april 4th to a US company
  - Used FreeBSD ever since
- Startup entrepreneur in internet or embeded technology
  - Most companies use Linux
- Currently co-owner of 6 companies
  - Datacentre, Cloud company, Web design buro

Why Ceph and ZFS

- Over the years used most storage types known to IT engineers
  - ApolloDomain tokenring, Hard/software raid, netapp, sun-clusters
- ZFS (2006) has been the easiest and greatest pleasure to use.
  - ZFS is becoming the FS for bigger systems in FreeBSD.
  - I'm using it since 2008, and it has never failed me.
- bhyve hypervisor (2011)
  - Runs most other Oses, including windows x64
- So the goal:
  - Running bhyve with RBD with Ceph on FreeBSD/ZFS

A bit about ZFS

- Copy on write FS
  - Never over-write an existing block
  - Filesystem is always consistent
  - State atomically advances at checkpoints
  - Metadata redundancy and data checksums
  - Selective data compression and deduplication
  - The fsync() system call is implemented by forcing a log write not by doing a checkpoint
- HAST is a 2 node “High Available” concept, bolted onto ZFS and CARP to create failover.
  - But seriously suffers from split-brain problems. And that is not for the faint of hart.

Porting to FreeBSD

- OS versions 9.3, 10.3, 11.0
  - 11.0 is going to be stable around sept this year
  - Code sludge has started
- Compilers
  - Clang is native, 3.4(10.3), 3.7, 3.8(11.0)
  - Gcc is native 4.2.1, pkgs: 4.6 upto 7
- Packaging system with 22.000 ports
  - Releases are not tied to OS releases
  - Packages are not always most current
  - Not all packages have upstreamed their BSD patches
    - Gtest/gmock
  - Using about 31 packages at the top level of Ceph
    - Resulting in about 500 dependancies

Most trivial stuff

- Compiler warnings:
  - You can switch them of, but do you really want to?

```
./log/Log.h:18:1: warning: class 'Entry' was previously declared
as a struct [-Wmismatched-tags]
class Entry;
^
./log/Entry.h:16:8: note: previous use is here
struct Entry {
       ^
./log/Log.h:18:1: note: did you mean struct here?
class Entry;
^~~~~
Struct
```

- Unused variables
  - Do you want to obfuscate code to prevent this, and add ifdef's around variable declaration as well.

Ignoring warnings

- This actually might discover a bug

```
    int r = -1;
#ifdef IPTOS_CLASS_CS6
    r = ::setsockopt(sd, IPPROTO_IP, IP_TOS, &iptos, sizeof(iptos));
#endif
#if defined(SO_PRIORITY)
  // setsockopt(IPTOS_CLASS_CS6) sets the priority of the socket as 0.
  // See http://goo.gl/QWhvsD and http://goo.gl/laTbjT
  // We need to call setsockopt(SO_PRIORITY) after it.
#if defined(__linux__)
    r = ::setsockopt(sd, SOL_SOCKET, SO_PRIORITY, &prio, sizeof(prio));
#endif
#endif
```

The trivial stuff

- Missing included files
  - Or different names for include files
- Mismatching defines
  - `#define MSG_MORE 0`
  - `#define O_DSYNC O_SYNC`
  - `#define ENODATA ENOATTR`
    - Turned out to be not so trivial.
- Mismatching system functions
  - `#define pthread_setname_np pthread_set_name_np`
- Missing functions
  - `pthread_getname_np()???`

    You need to go fishing in kernel space to get it out, according hackers@freebsd.org.

    So delayed for later on.

The simple stuff

- Remember??
  - `#define ENODATA ENOATTR`
- This runs in

```
./os/filestore/chain_xattr.h
   do {
      get_raw_xattr_name(name, i, raw_name, sizeof(raw_name));
      r = sys_fremovexattr(fd, raw_name);
    } while (r != -ENODATA);
  }
```

- Don’t include `boost-includes` earlier:

```
boost/cerrno.hpp:#define ENODATA 9919
```

How about CLOCK_* ??

- Different types of solutions used.
  - In compat.h:

```
#if !defined(CLOCK_MONOTONIC_COARSE)
#if defined(CLOCK_MONOTONIC_FAST)
#define CLOCK_MONOTONIC_COARSE CLOCK_MONOTONIC_FAST
#else
#define CLOCK_MONOTONIC_COARSE CLOCK_MONOTONIC
#endif
#endif
```

POSIX stuff

- Lots of pthread stuff
- Was this an optimization, or a bug fix?

```
#if defined(PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP)
if (prioritize_write) {
  pthread_rwlockattr_t attr;
  pthread_rwlockattr_init(&attr);
  // Setting the lock kind to this avoids writer starvation as long
  // as long as any read locking is not done in a recursive fashion.
  pthread_rwlockattr_setkind_np(&attr,
      PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP);
  pthread_rwlock_init(&L, &attr);
}
#endif
```

Semantics

- wait_until(lock, time)
- Now what if time is in the past.

```
auto start = std::chrono::system_clock::now();
delay = _get_delay(c);
while (((start + delay) > std::chrono::system_clock::now()) ||
       !((max == 0) || (current == 0) || ((current + c) <= max))) {
  (*ticket)->wait_until(l, start + delay);
  delay = _get_delay(c);
}
```

- Change to

```
while (true) {
  if (!((max == 0) || (current == 0) || (current + c) <= max)) {
    (*ticket)->wait(l);
  } else if (delay > std::chrono::duration<double>(0)) {
    (*ticket)->wait_for(l, delay);
  } else {
    break;
  }
```

First time right??

- Now the unittest fails:

```
stop = true;
for (auto &&i: gts) i.join();
gts.clear();
for (auto &&i: pts) i.join();
pts.clear();
```

- Termination of the throttle needs more care

```
getter_stop = true; milliwait(100);
c.notify_all();
for (auto &&i: gts) i.join();
gts.clear();

putter_stop = true; milliwait(100);
c.notify_all();
for (auto &&i: pts) i.join();
pts.clear();
```

Semantics 2

extattr_list_file() returns a list of attributes present in the requested namespace. Each list entry consists of a single byte containing the length of the attribute name, followed by the attribute name. The attribute name is not terminated by ASCII 0 (nul).

- What it does not say is that the order attributes are returned, are in the same order they are inserted!

  ::memcmp(actual, buffer, buffer_size)

  Does not need to be true !!

Testing ??

Thank you for such a nice set of tests !!!

Testing ???!!!!

- Make check.
  - Unittests
    - Great help to assert that most things work
      - Debug reporting is minimal, little help if things go wrong.
    - Some are really slow, doing benchmarking??
      - Env setting to disable?
  - Scripts
    - No description of what the test does.
    - Small ones, big ones in *.sh(bash), *.py, cython, nosetests
    - Not always matching up with autoconfig (RBD)
    - Lots of other tools used: grep, sed, [ ], perl
    - But also convert and jq, jq is used only in 1 script...

Running complex tests

- Why would I need 2 signals to reminate ceph-{osd,mon}??
  - And osd-markdown.sh kills the it on the first hit?
- I'd like to keep the data and logs for post-mortum analysis, if things go bad.
- If OSDs die, we still continue trying to test rados and rbd access.
  - That blocks the scripts and then takes forever to complete

Number of changes

Total ifdef's CEPH_* HAVE_* cplusplus	1867 724 185 127	Not counting Rocksdb and Gtest/Gmock
__linux__ OS_LINUX	74 104	Used for Rocksdb, gtest
__FreeBSD__ OS_FREEBSD	61 19	
DARWIN __APPLE__	37 14	
AIX __sun	9 10	

Compat.h has 115 lines, including header

Recommendations

- All “trivial” OS mismatches should go into:
  - `“include/compat.h”`
    - And should be included first or at least before any other boost includes
- Conditionalize linux-isms
  - And generate warnings (or errors) during running if not fixed for other Oses
    - lsb_release, hdparm, gpart, …
    - And an indicator of sorts would be nice
-

Recommendations

- Scripts should us as much std-shell as possible
  - Prevent serious bash-ism if you can
  - Don't use sed to replace the last (empty) line with new data.
    - A HERE-file works way much more ledgible.
- Cleanup after a test: tmp-files, *logs, cores….
  - Perhaps are cores after GTEST_DEATH a typical FreeBSD problem.

Things to do……

- Teuthology integration??
- ZFS integration
- RBD in userspace for bhyve
- AIO compatibility layer for BlueStore
- Ceph-deploy

- More urgent things?

Questions