Testing Ceph: Status, Development, & Opportunities

Greg Farnum – Cephalocon – May 20, 2019
Teuthology: Study of Cephalopods
Teuthology: Early Design
Eight Years Ago...

- We needed to formalize Ceph testing
- Nothing existed for distributed systems
  - Our first try involved Autotest. It was unsuccessful.
- We really needed to be able to manipulate multiple machines at once
The “Orchestra” Module

Python
+ Paramiko (SSH)
+ gevent
= orchestra

• Real-time
• Interactive
• Central controller
• Full SSH protocol
  - (channels!)
• Not Chef
• Not Fabric

ccluster = Cluster(...)
ccluster.run(...)
ccluster.only('x86').run(...)
ccluster.exclude('x86').run(...)

Teuthology started as a test runner

Run tasks on targets as told to by roles

Automatically:

- Setup
- Monitor health
- Run test(s)
- Archive results
- Archive logs, core dumps, etc
- Clean up
Teuthology: Targets

targets:
- ubuntu@sepiaXX.ceph.dreamhost.com
- ubuntu@sepiaYY.ceph.dreamhost.com
- ubuntu@sepiaZZ.ceph.dreamhost.com
Teuthology: Roles

roles:
- [mon.0, mds.0, osd.0]
- [mon.1, osd.1]
- [mon.2, client.0]

targets:
- ubuntu@sepiaXX...
- ubuntu@sepiaYY...
- ubuntu@sepiaZZ...
Teuthology: Tasks

tasks:
- ceph:
- kclient: [client.0]
- workunit:
  clients:
    all:
      - suites/dbench.sh
Teuthology tasks can be context managers

tasks:
  - ceph:
  - kclient: ...
  - autotest: ...
  - interactive:
Teuthology: Combine yaml fragments

Automatically combines individual yaml files into single test

~$ teuthology targets.yaml
kclient_roles.yaml kclient_dbench.yaml
--archive kernel_dbench_test.logs
targets:
- ubuntu@seapiaXX.ceph.dreamhost.com
- ubuntu@seapiaYY.ceph.dreamhost.com
- ubuntu@seapiaZZ.ceph.dreamhost.com

roles:
- [mon.0, mds.0, osd.0]
- [mon.1, osd.1]
- [mon.2, client.0]

tasks:
- ceph:
- kclient: [client.0]
- workunit:
  clients:
    all:
      - suites/dbench.sh
Teuthology logs all job output

2011-06-21T10:00:44/
├── ceph-sha1
├── config.yaml
├── remote
│   ├── ubuntu@sepia70.ceph.dreamhost.com
│   │   └── log
│   │       ├── client.admin.log.bz2
│   │       └── mds.0.log.bz2
│   │   └── mon.0.log.bz2
│   └── osd.0.log.bz2
│       └── syslog
│           ├── kern.log.bz2
│           └── misc.log.bz2
│   └── ubuntu@sepia71.ceph.dreamhost.com ...
│       └── ubuntu@sepia72.ceph.dreamhost.com
│           └── autotest
│               └── ...
│   └── log ...
│       └── syslog ...
└── summary.yaml
    └── teuthology.log
Teuthology: The Sepia lab

- A community lab devoted to Ceph development
- ~200 SSD-based “smithi” nodes
- ~50 HDD-based “mira” nodes
- Devoted to running Ceph tests
- SSH access granted to engaged developers (>100 people)
Teuthology: Locking servers

- Don’t want to collide targets with other developers — lock the nodes!

```
$ teuthology-lock --lock-many 3 -m smithi

targets:

smithi006.front.sepia.ceph.com: ssh-rsa AAAAB3Nz...ROWBz
smithi060.front.sepia.ceph.com: ssh-rsa AAAAB3Nz...UcfMb
smithi084.front.sepia.ceph.com: ssh-rsa AAAAB3Nz...Av0UT
```
Teuthology: teuthology-schedule & beanstalkd

tequhtology-schedule targets.yaml kclient_roles.yaml kclient_dbench.yaml

• Puts jobs into beanstalkd

• Pulled off sequentially[1] by running teuthology-workers that lock required nodes, execute job, store results
  
  - [1] modulo the priority -p flag, which orders lower-numbered priorities earlier than higher-numbered ones in the queue
Ceph QA “Suites”

- “Suites” are directories of yaml fragments that get combined to form complete jobs
Ceph QA “Suites”

~src/ceph [master]$ ls qa/suites/rados/verify/*
qa/suites/rados/verify/ceph.yaml  qa/suites/rados/verify/rados.yaml

qa/suites/rados/verify/clusters:
+ fixed-2.yaml  openstack.yaml

qa/suites/rados/verify/d-thrash:
default none.yaml

qa/suites/rados/verify/objectstore:
bluestore-bitmap.yaml  bluestore-comp.yaml  bluestore.yaml  filestore-xfs.yaml

qa/suites/rados/verify/tasks:
mon_recovery.yaml  rados_api_tests.yaml  rados_cls_all.yaml
Ceph QA “Suites”

```
~/src/ceph [master]$ ls qa/suites/rados/verify/*
qa/suites/rados/verify/ceph.yaml  qa/suites/rados/verify/rados.yaml

qa/suites/rados/verify/clusters:
+ fixed-2.yaml  openstack.yaml

qa/suites/rados/verify/d-thrash:
default  none.yaml

qa/suites/rados/verify/objectstore:
bluestore-bitmap.yaml  bluestore-comp.yaml  bluestore.yaml  filestore-xfs.yaml

qa/suites/rados/verify/tasks:
mon_recovery.yaml  rados_api_tests.yaml  rados_cls_all.yaml
```
Ceph QA “Suites”

```bash
~ /src/ceph [master]$ ls qa/suites/rados/verify/*
qa/suites/rados/verify/ceph.yaml  qa/suites/rados/verify/rados.yaml

qa/suites/rados/verify/clusters:
+ fixed-2.yaml  openstack.yaml

qa/suites/rados/verify/d-thrash:
default none.yaml

qa/suites/rados/verify/objectstore:
bluestore-bitmap.yaml  bluestore-comp.yaml bluestore.yaml  filestore-xfs.yaml

qa/suites/rados/verify/tasks:
mon_recovery.yaml  rados_api_tests.yaml  rados_cls_all.yaml
```
Ceph QA “Suites”

~src/ceph [master]$ ls qa/suites/rados/verify/*

qa/suites/rados/verify/ceph.yaml  qa/suites/rados/verify/rados.yaml

qa/suites/rados/verify/clusters:
+ fixed-2.yaml  openstack.yaml

qa/suites/rados/verify/d-thrash:
  default  none.yaml

qa/suites/rados/verify/objectstore:
  bluestore-bitmap.yaml  bluestore-comp.yaml  bluestore.yaml  filestore-xfs.yaml

qa/suites/rados/verify/tasks:
  mon_recovery.yaml  rados_api_tests.yaml  rados_cls_all.yaml
Ceph QA “Suites”

~/src/ceph [master]$ ls qa/suites/rados/verify/*

qa/suites/rados/verify/ceph.yaml   qa/suites/rados/verify/rados.yaml

qa/suites/rados/verify/clusters:
+ fixed-2.yaml  openstack.yaml

qa/suites/rados/verify/d-thrash:

default  none.yaml

qa/suites/rados/verify/objectstore:

bluestore-bitmap.yaml  bluestore-comp.yaml  bluestore.yaml  filestore-xfs.yaml

qa/suites/rados/verify/tasks:

mon_recovery.yaml  rados_api_tests.yaml  rados_cls_all.yaml
Ceph QA Suites: Test Coverage

- big
- buildpackages
- cephmetrics
- dummy
- experimental
- fs
- hadoop
- kcephfs
- knfs
- krbd
- marginal
- multimds
- powercycle
- rados
- rbd
- rgw
- samba
- smoke
- stress
- teuthology
- tgt
- upgrade
Ceph QA Suites: Thrashers

- Thrashers are chaos monkeys that we run while other tests are in progress

```python
ceph/qa/tasks/ceph_manager.py "Thrasher" class
tasks:
- thrashosds:
  timeout: 1200
  chance_pgnum_grow: 1
  chance_pgpgnum_fix: 1
```
Teuthology: How We Use It
Ceph Branch Testing

- Committed developers are granted access to the sepia lab; may run tests on Work-In-Progress branches
  - 148 SSH keys are in the repository, though some are bots/stale
- Push a branch to ceph-ci.git
  teuthology-suite -m mira -c wip-greg-12345-test -s rados
- Results publicly available at http://pulpito.ceph.com
Ceph Pull Request Testing

• Tech leads and reviewers review PRs, build integration branches, and run them through the appropriate suite(s) to check for issues
  – ptl-tool.py to build integration branch, push to ceph-ci.git, merge PRs after auditing results
  – NOTHING merges to Ceph master without passing tests!

• Results publicly available at http://pulpito.ceph.com
Ceph Nightlies

- Suites are scheduled via crontab
- Generally run from 1 to 7 times weekly
- Results publicly available at [http://pulpito.ceph.com](http://pulpito.ceph.com)
Testing Ceph: Other frameworks
Jenkins tests

- ceph-ansible and ceph-volume have their own tests (tox/Jenkins)
  - http://docs.ceph.com/ceph-ansible/master/testing/index.html
- unit tests run in our “make check” build target
  - Important: “ceph-object-corpus”
- These are all automatic on every PR
Teuthology: Test Gaps
Test Coverage Successes

- Functional coverage is actually quite good!
- Write specific tools such as ceph_test_rados to validate API behavior, run against thrashers
- Deliberately inject failures to test daemon recovery
  - eg qa/suites/rados/thrash/msgr-failures/osd-delay.yaml
- Fiddle with raw objects and test CephFS recovery
- Fiddle with disk state and test OSD/RADOS recovery
Test Coverage Gaps

- Teuthology handles daemons directly by sshing into test nodes
  - It does not test our systemd (or other init system) units :(
- Performance testing
  - New “perf” suite uses the Ceph Benchmark Tool, but limited tests and no analysis
- Scale testing: Tests are expected to run within hours, in as little space as possible
- Teuthology handles Ceph installs and cluster config
  - It does not test deployment tools (ceph-ansible, ceph-deploy)
Test Coverage Fixes

- Problem: Teuthology handles daemons directly by sshing into test nodes

- Proposed Solution: Expand the API so we can restart and signal using the init system instead of

```python
mon.run(args=['killall', '-9', 'rados'], wait=True, check_status=False)
```
Test Coverage Fixes

- Problem: Performance testing

- First-pass solution: Gather historical data on performance, then guess at a rough outer limit and fail tests when it exceeds X% difference

- Proposed Long-term Solution: Write a new service to track performance numbers and alert when they are out of bounds?
Test Coverage Fixes

- Problem: Scale testing: Tests are expected to run within hours, in as little space as possible

- Decision: an integration test framework is not a good place for long-running tests.

- Proposed Solution: determine how to mock up and inject “aged” and large-scale data within short tests

- Proposed Solution: design tests that run with longer-term allocations in their own lab area
Test Coverage Fixes

- Problem: Teuthology handles Ceph installs and cluster configuration

- Proposed Solution: Write a new API so that we can request installs and farm them out to ceph-ansible, DeepSea, Rook, etc
Test Coverage Fixes

• Problem: New ceph-mgr *orchestrators* are not integrated into testing
  
  – These rely on ceph-ansible/DeepSea/Rook to do things like deploy new Ceph daemons

• Solution: In progress orchestrator tests built on tox/terraform/Jenkins (by Sebastian Wagner)

• Proposed Solution: Hurry up and integrate these deployment tools into teuthology framework

• Note: This is a key example of how non-integrated testing can have long-term negative consequences
Teuthology: Framework Weaknesses
Problem: Strongly tied to sepia

- It’s supposed to deploy anywhere, but...
  - Hard-coded values sneak in
  - Documentation often unclear, with many services to wrangle
- Other groups running teuthology do a local port
- Makes developing tests difficult — the code-test-debug cycle involves locking in-demand lab machines, doing machine images & package installs!
- You need built Ceph packages — no testing local source code
- Makes it hard for third parties to contribute to mainline teuthology development
- Makes it hard for new Ceph contributors to test their patches
Solutions: Strongly tied to sepia

- Should have: made regular teardown and setup a part of our practice
- Should have: worked more closely on it when we discovered groups having trouble with setup and install
- Should have: considered non-lab users more carefully when making some changes, such as to require packages
- Previously: “teuthology-openstack”
  - set of scripts that would deploy a teuthology instance and run a test suite within OpenStack clouds. Active in at least one group but died upstream due to lack of maintenance/interest
Solutions: Strongly tied to sepiap

- **Now:** “vstart-runner”
  - A restricted API that lets you run some tests out of a local source tree build, in addition to within a full teuthology lab
- **Now:** Invest work to foster community
  - “Ceph Testing Weekly” meeting to discuss https://ceph.com/testing/
  - Reach out to known users
  - Active work to merge their enhancements and fixes upstream
- **Future:** Look at “localization” patches people have written and replace them with configuration options to make new deployments easier
- **Future:** Rebuild teuthology-openstack with newer, reliable APIs
Solutions: Strongly tied to sepia

• Success: SUSE has almost finished bringing their teuthology-openstack fork back to upstream (Kyrylo Shatskyy)

• Success: Digital Ocean deploys teuthology in their cloud github.com/digitalocean/digitalocean-ceph-lab (Adam Wolfe Gordon)
Problem: Lacks trend analysis tools

- Robust mechanisms for failing individual tests
  - Presence of core dumps, WARN or ERR log messages, failed test conditions, etc
- Easy to look at individual test run pass/fail at http://pulpito.ceph.com
- But no automatic identification of trends:
  - Test pass rate
  - Test execution time
  - Performance numbers
Solutions: Lacks trend analysis tools

- Future: build (or deploy existing) trend-tracking software with Pulpito?
Problem: Suites explode in size

124310 jobs (not yet filtered)

- No way to prioritize tests
  - Knowing which failures to analyze
  - Get faster feedback while suite execution proceeds
  - Help resource-constrained users run the most important tests
Ceph QA Suites: Subset Functionality

- Suites geometric combinations have gotten VERY large


- “subset” functionality uses all YAML fragments, but not the total combinatoric explosion (until you run every subset within your denominator)

  2018-08-01 05:10:09,065.065 INFO:teuthology.suite.build_matrix:Subset=1/500

Solutions: Suites explode in size

124310 jobs (not yet filtered)

- Done: “subset” scheduling option
- Done: “filter” scheduling option that matches regex pattern in test “name”
- Done: “rerun” scheduling option runs only jobs that failed in a previous run
Problem: Scheduling is primitive

- Jobs are picked off beanstalkd, then their executor (1 of 50) tries to lock servers
  - A 5-node job never runs, because the 2-node jobs always beat it to locking machines!
- Scheduled nightlies can be delayed for days, then run a full suite on a week-old sha1
Solutions: Scheduling is primitive

- Workaround hack: make all jobs run in same number of nodes
- Workaround hack: Monitor beanstalkd queue, kill and reschedule old nightly scheduled suites
- Proposed Solution: Update existing scheduled test runner: lock needed machines before removing jobs from queue, so tests don’t race each other for resources
- Proposed Solution: Write or find a more robust scheduler
- Wants: Automatically adjust number of “nightly” jobs based on “manual” suites in queue, time since last suite pass, etc
  - Select the sha1 of code to run at execute time instead of at schedule time
Modernizing Teuthology
Comprehensive Changes Coming

- Kubernetes and Rook will cause big changes
- Actively discussing how to test
- Run Kubernetes inside teuthology
  - MiniKube? Somebody’s “real” install scripts?
- Proposed Solution: Implement “deployer” interfaces so tests can be agnostic to new environments
Write Kubernetes tests instead?

- Kubernetes and Rook will cause big changes

- Actively discussing how to test

- Write Ceph tests in a Kubernetes-affiliated framework (Prow!)
  - Focus on integration points
  - Can we identify areas of risk well enough without reimplementing all existing tests?
Problem: Physical machines are first-class citizens

- Things like package installation apply to our targets
  - Also narrower issues: some suites powercycle
- Daemons belong to machines too
- But for the world of Kubernetes that is nonsensical; packages belong to containers!
Solutions: Physical machines are first-class citizens

- Proposed Solution: Define new interfaces for making packages available and starting them
The End!
FOR MORE INFORMATION

- https://github.com/ceph/teuthology
- https://github.com/ceph/ceph/tree/master/qa
- http://docs.ceph.com/teuthology/docs
- #sepia on irc.oftc.net
- https://ceph.com/testing/
THANK YOU!

Greg Farnum

gfarnum@redhat.com
@gregsfortytwo