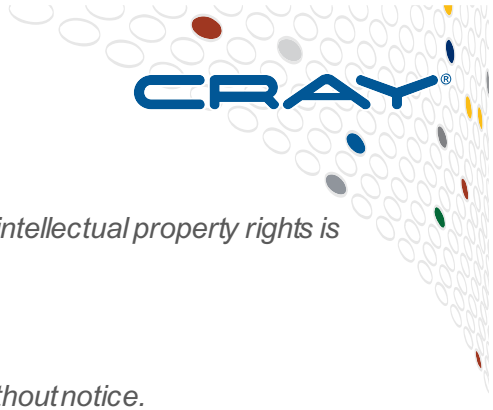
Lustre* Testing: The Basics

Justin Miller, Cray Inc.
James Nunez, Intel Corporation
LAD '15
Paris, France

Legal Disclaimer



Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Intel Legal Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Agenda

Motivation and Goals

Relevance of Lustre Test Suites

Who Should Care and Why

Overview and Terminology

Coverage

Configuration and Execution

Auster and Examples

What to do in Case of Error

Motivation and Goals

Increase participation in Lustre testing

- Low barrier to contributing
- Testing your system is valuable for you and the community
- Improving the upstream release helps everyone

Document and demonstrate the Lustre Tests Suites

- Cover the basics on how to get started
- Tell you what to run and how to build up from there
- Continue the conversation on how to improve

Why Should I Care About the LTS?

Administrators

- Verify your software and hardware configuration before production
- Collect baseline performance data to compare against later

Developers

- Test outside the area of code directly modified, interaction of features
- Know before you submit if your patch will pass upstream testing

Everyone

- Improve quality and stability by catching regressions, validating functionality, and testing interaction of features with dynamic tests

Lustre Test Suites Relevance

Lustre Test Suites (LTS)

- largest collection of Lustre specific tests
- have evolved and expanded with every release

Part of the review process

- Every patch submitted to code review goes through automated testing
  - Multiple configurations for proposed changes
  - More extensive changing for change once it is committed

There are issues, some minor and some major

- See Nathan Rutman's presentation from LAD '12

LTS Overview

What does LTS provide?

- Over 1600 tests organized by purpose and function
- Bash scripts, C programs, and external applications
- Utilities to create, start, stop, execute tests

What can you do with LTS?

- Execute the test process automatically, or with discreet steps
- Run individual tests, test suites, or groups of suites
- Experiment with configurations and features: ldiskfs, zfs, DNE, HSM, …

Terminology

Lustre Test Suites (LTS)

- Everything in lustre-(client-)tests-*.rpm
  - lustre-iokit.rpm contains complimentary scripts and programs

Test Suite

- In /usr/lib64/lustre/tests there are individual suites of tests, e.g. sanity.sh

Individual Test

- A suite is composed of individual tests, e.g. test_17n

Test Group

- Suites can be bundled into a group for back-to-back execution, e.g. regression

Coverage

LTS Examples

- Regression – sanity, sanityn
- Feature Specific – sanity-hsm, sanity-lfsck, ost-pools
- Configuration - conf-sanity
- Recovery and Failures – recovery-small, replay-ost-single
- Functional – parallel-scale-nfsv4, mmp
- Baseline Performance – sanity-benchmark, obdfilter-survey
- Dynamic – racer

Levels of Testing

- Range of tests from isolated components to entire file system
- Probing into core of Lustre - fail_loc to induce a particular behavior

Things You Really Need to Know

Individual tests can reformat server targets

LTS can fail due to issues outside of tests and Lustre

Documentation of individual tests is lacking - in scripts and outside.

Hierarchy of test importance - There is no order of importance in the test scripts.

Always use the latest version of LTS

Requirements

Environment

- Normal Lustre Installation
- Resolvable host names
- Passwordless ssh with no prompt for all-to-all

Software

- Install lustre-(client-)tests-*.rpm and lustre-iokit.rpm
- Latest e2fsprogs
- Recommended utilities: pdsh
- Optional: IOZone, bonnie++, pios, dbench, MPI, posix-1.0-wc1

Configuration

LTS make extensive use of environment variables

Configuration Files

- A simple configuration will allow you to run most of the tests
- Specific feature tests require more configuration
- Recommended: Same configuration file on all nodes

Variables

- Defined in cfg/local.sh and cfg/ncli.sh
- MDS/MDT and OSS/OST
- PDSH (-S return code)
- RCLIENTS

Execute LTS with Auster

Auster

- Create/Execute/Clean up/Log collection
- Output/results
- Run a single test from a test suite or run a group of tests multiple times
- Some tests are excluded

Running LTS

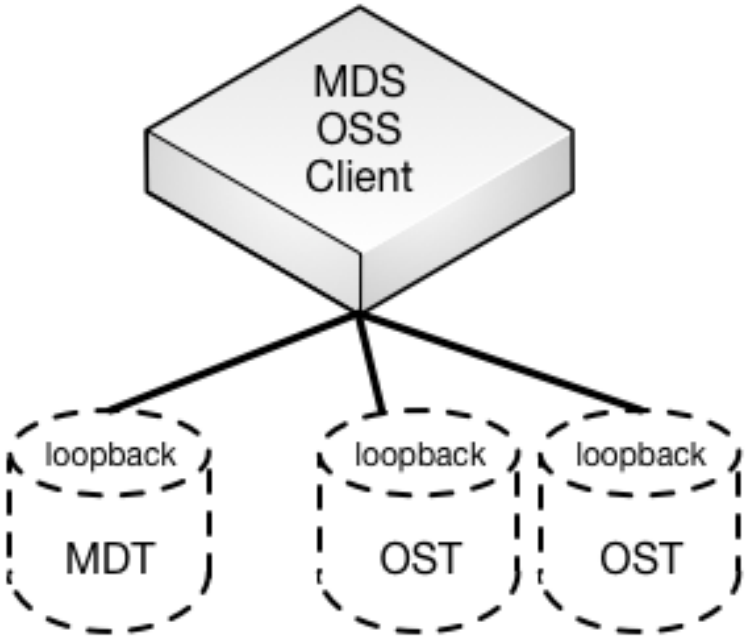
- Example: /usr/lib64/lustre/tests/auster -f multinode -rv sanity --only 33d

Configuration Example – Default

cfg file: local.sh

Command:

./auster -v sanity --only 0

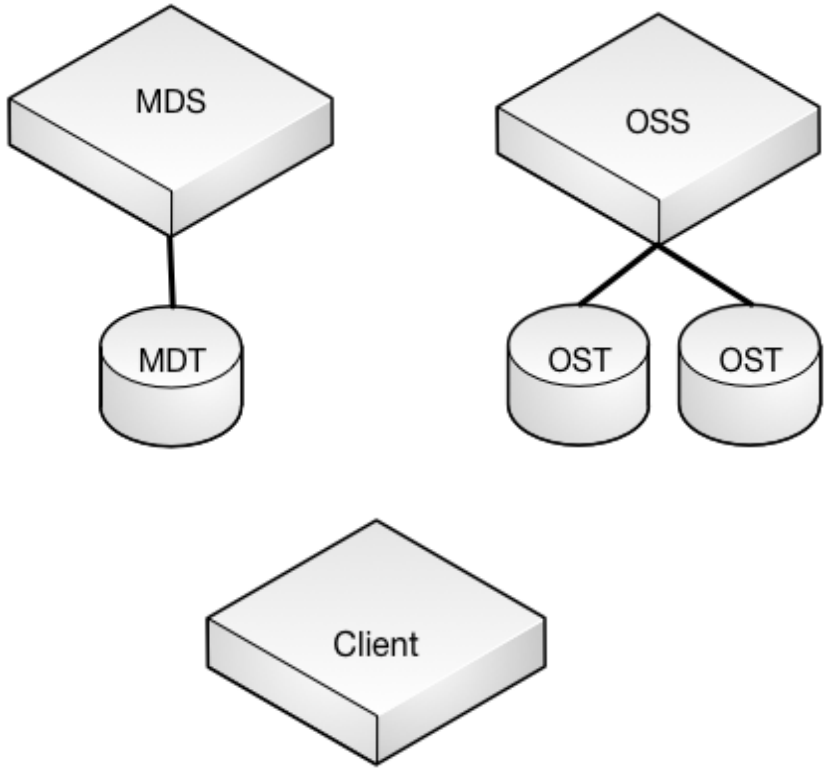


Configuration Example – Simple

cfg file: cfg/cfg_smpl.sh
mds_HOST=node01
MDSDEV1=/dev/vda3
OSTCOUNT=2
ost_HOST=node02
OSTDEV1=/dev/vda3
ost2_HOST=node02
OSTDEV2=/dev/vda4

PDSH="pdsh -S -Rssh -w”
# Source local.sh
. $LUSTRE/tests/cfg/local.sh

Command:
./auster -v -r -f cfg_smpl sanity

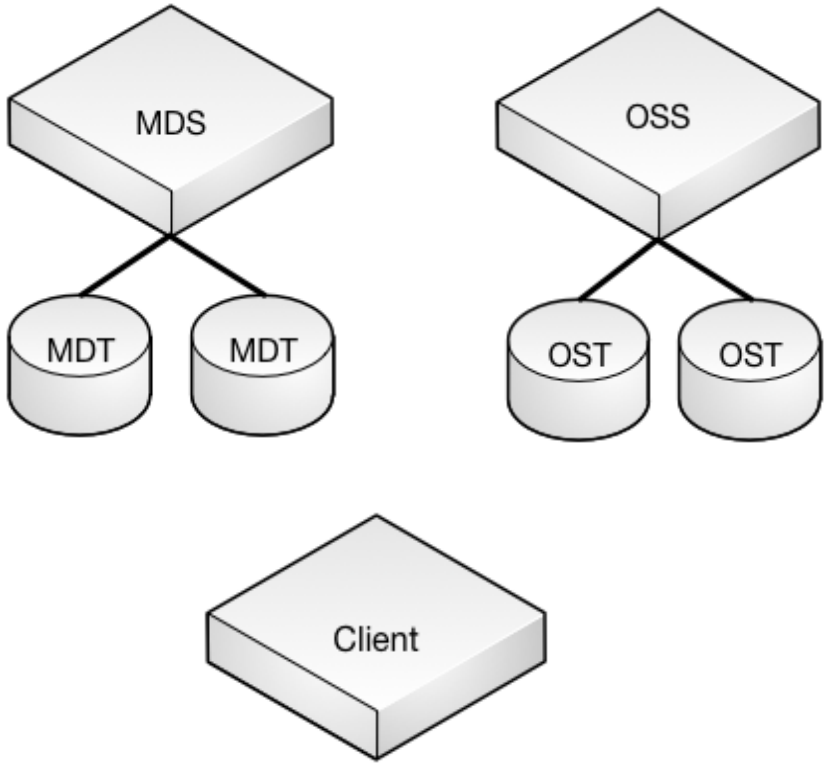


Configuration Example – DNE

cfg file: cfg/cfg_dne.sh
MDSCOUNT=2
mds_HOST=node01
MDSDEV1=/dev/vda3
mds2_HOST=node01
MDSDEV2=/dev/vda4
OSTCOUNT=2
ost_HOST=node02
OSTDEV1=/dev/vda3
ost2_HOST=node02
OSTDEV2=/dev/vda4

PDSH="pdsh -S -Rssh -w"
. $LUSTRE/tests/cfg/local.sh

Command:
./auster -v -r -f cfg_dne sanity



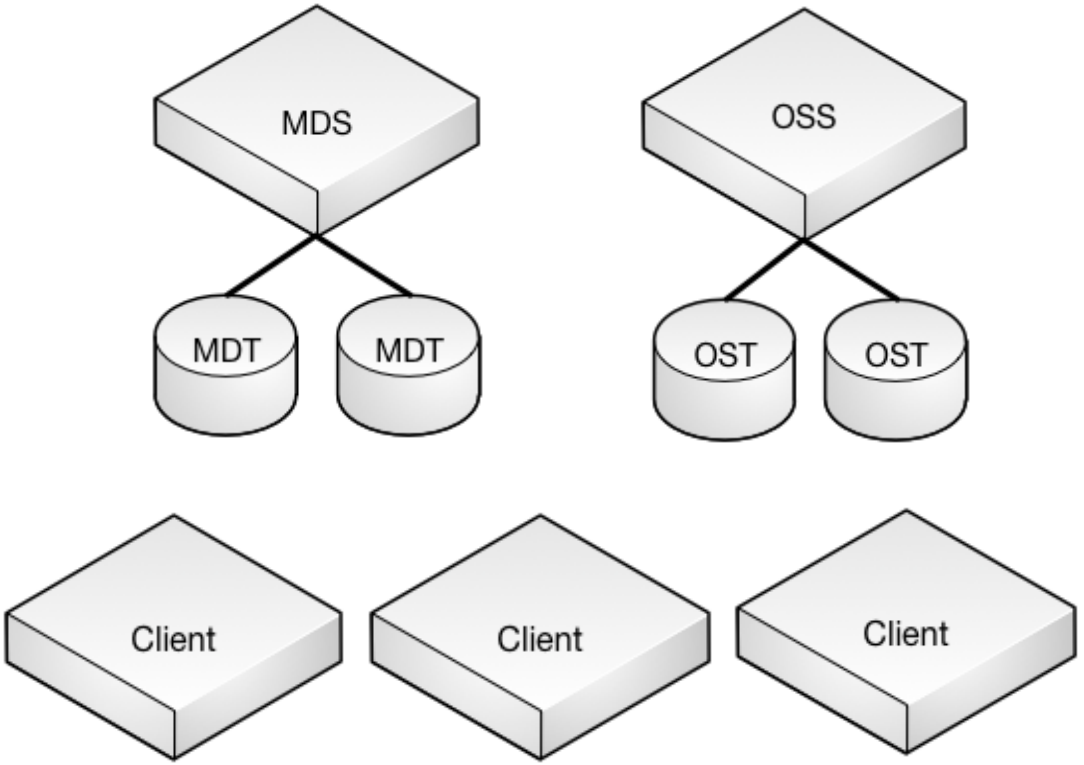
Configuration Example – Multiple Clients

cfg file: cfg/cfg_multi.sh
MDSCOUNT=2
…
PDSH="pdsh -S -Rssh -w"

RCLIENTS="node03 node04 node05"

. $LUSTRE/tests/cfg/ncli.sh

Command:
./auster -v -r -f cfg_multi sanity



Configuration Example – HSM

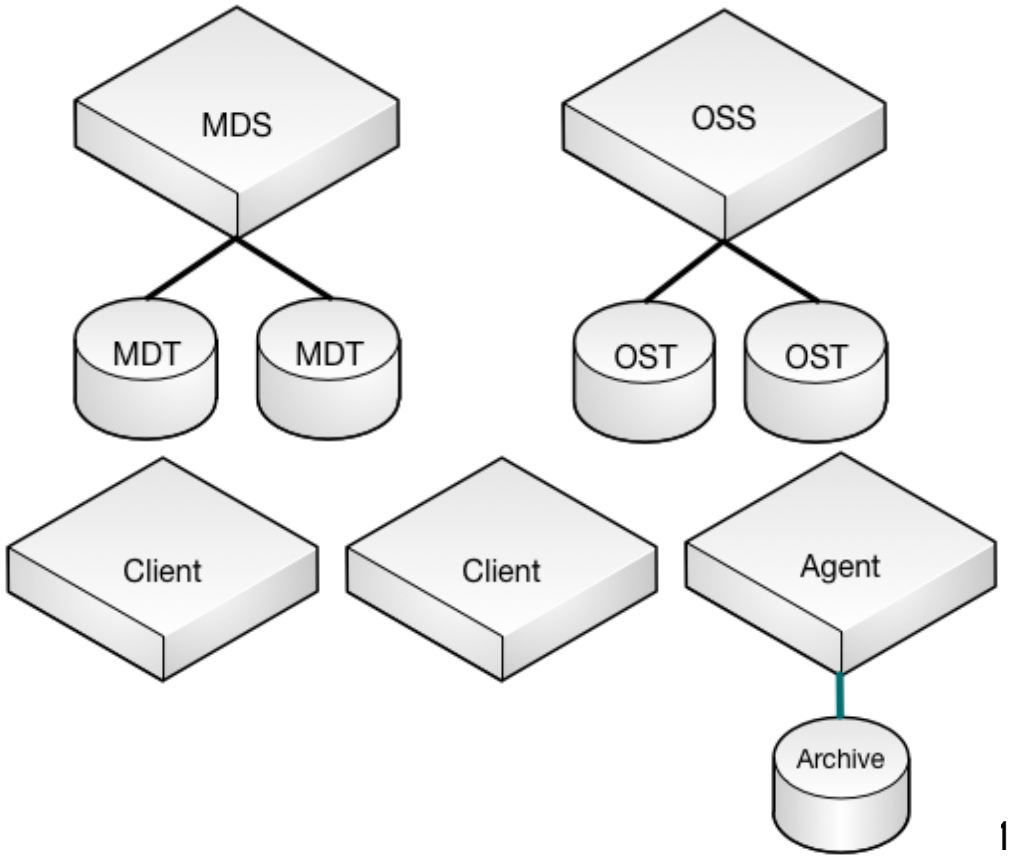
cfg file: cfg/cfg_hsm.sh
MDSCOUNT=2
…
PDSH="pdsh -S -Rssh -w"

RCLIENTS="node03 node04 node05"

SHARED_DIRECTORY=/scratch
AGTDEV1=/archive
agt1_HOST=node05
HSMTOOL_VERBOSE="-v -v -v -v -v -v"

. $LUSTRE/tests/cfg/ncli.sh

Command:
./auster -v -f cfg_hsm -r sanity-hsm



Output & Results

Individual test and suite status - PASS/FAIL/SKIP
Auster output

- Results
- Node information for each host
- Test suite logs – stdout, dmesg and console
- Test failures
  - sanity.test_235.debug_log.<node>.<time>.log
  - sanity.test_235.dmesg.<node>.<time>.log
  - debug_log and dmesg logs will be gathered to the node executing the tests from all nodes in the configuration

Auster Results – Test Suite

```
--== acceptance-small: conf-sanity ==-- Sat Aug 22 14:03:11
PDT 2015
Running: bash /usr/lib64/lustre/tests/conf-sanity.sh
excepting tests: 32newtarball 59 64
skipping tests SLOW=no: 30a 31 45 69
Stopping clients: node18,node17 /lustre/archive (opts:)
Stopping client node17 /lustre/archive opts:
…
== conf-sanity test 0: single mount setup == 14:05:10
(1440277510)
start mds service on mds02
Starting mds1:   /dev/sdb /lustre/archive/mdt0
Started archive-MDT0000
…
pdsh@node18: mds02: ssh exited with exit code 1
modules unloaded.
Resetting fail_loc on all nodes...done.
PASS 0 (38s)

== conf-sanity test 1: start up ost twice (should return errors)
== 14:05:48 (1440277548)
…
== conf-sanity test 5f: mds down, cleanup after failed mount
(bug 2712) == 14:15:26 (1440278126)
Loading modules from /usr/lib64/lustre
detected 8 online CPUs by sysfs
libcfs will create CPU partition based on online CPUs
debug=-1
subsystem_debug=all -lnet -lnd -pinger
../lnet/lnet/lnet options: 'networks=o2ib(ib0) accept=all'
gss/krb5 is not supported

 SKIP: conf-sanity test_5f combined mgs and mds
Resetting fail_loc on all nodes...done.
SKIP 5f (11s)

== conf-sanity test 6: manual umount, then mount again ==
14:15:37 (1440278137)
start mds service on mds02
…
```

Test Groups

Run multiple test suites at once

- auster –v –r –f local –g regression

Regression

- Location: lustre/tests/test-groups
- Contains: runtests, sanity, sanityn, sanity-benchmark, metadata-updates, racer, lnet-selftest, replay-single, conf-sanity, recovery-small, replay-ost-single, replay-dual, replay-vbr, insanity, sanity-quota, sanity-sec, sanity-gss, lustre-rsync-test, ost-pools, mmp, obdfilter-survey, sgpdd-survey, sanity-scrub

Test Groups (cont)

Common test groups

- review-ldiskfs - lnet-selftest, sanity-scrub, sanity-sec, sanity
- review-zfs-part-1: sanity-hsm, sanity-lfsck, ost-pools, sanity-quota, sanityn, sanity, runtests
- review-zfs-part-2: lustre-rsync-test, insanity, replay-ost-single, recovery-small, conf-sanity, replay-single
- review-dne-part-1:lustre-rsync-test, recovery-small, conf-sanity, sanityn, sanity
- review-dne-part-2: sanity-sec, insanity, replay-ost-single, ost-pools, sanity-hsm, sanity-lfsck, sanity-scrub, sanity-quota, replay-single, mmp, runtests

Auster Results – Test Group

```
TestGroup:
   test_group: regression
   testhost: node18
   submission: Sat Aug 22 09:22:13 PDT 2015
   user_name: root

Tests:
-
      name: runtests
      description: auster runtests
      submission: Sat Aug 22 09:22:13 PDT 2015
      report_version: 2
      SubTests:
      -
         name: test_1
         status: PASS
         duration: 210
         return_code: 0
         error:
      duration: 216
      status: PASS
…
```

```
…
-
      name: test_110f
      status: SKIP
      duration: 1
      return_code: 0
      error: "needs\ \>=\ 2\ MDTs"
-
      name: test_111
      status: PASS
      duration: 18
      return_code: 0
      error:
-
      name: test_113
      status: PASS
      duration: 61
      return_code: 0
      error:
duration: 2307
status: PASS
```

Options - llmount

Auster

- Set up, format, run tests, and cleanup Lustre file system

llmount

- Set up and format Lustre file system

  NAME=cfg_dne llmount.sh

- Execute tests on command line

  NAME=simple ONLY=0 sanity.sh

llmountcleanup

- Cleanup Lustre file system

  NAME=cfg_dne llmountcleanup.sh

Test Failures

Check system components

Environment variables

Verbose output

- VERBOSE=true
- bash -x sanity.sh

Run outside auster

- NAME=<cfg_file> ONLY=0a ./sanity.sh

Try simple single test

- auster –v –f <cfg_file> –r sanity --only 0a

Some test fail by design; only seen in output logs

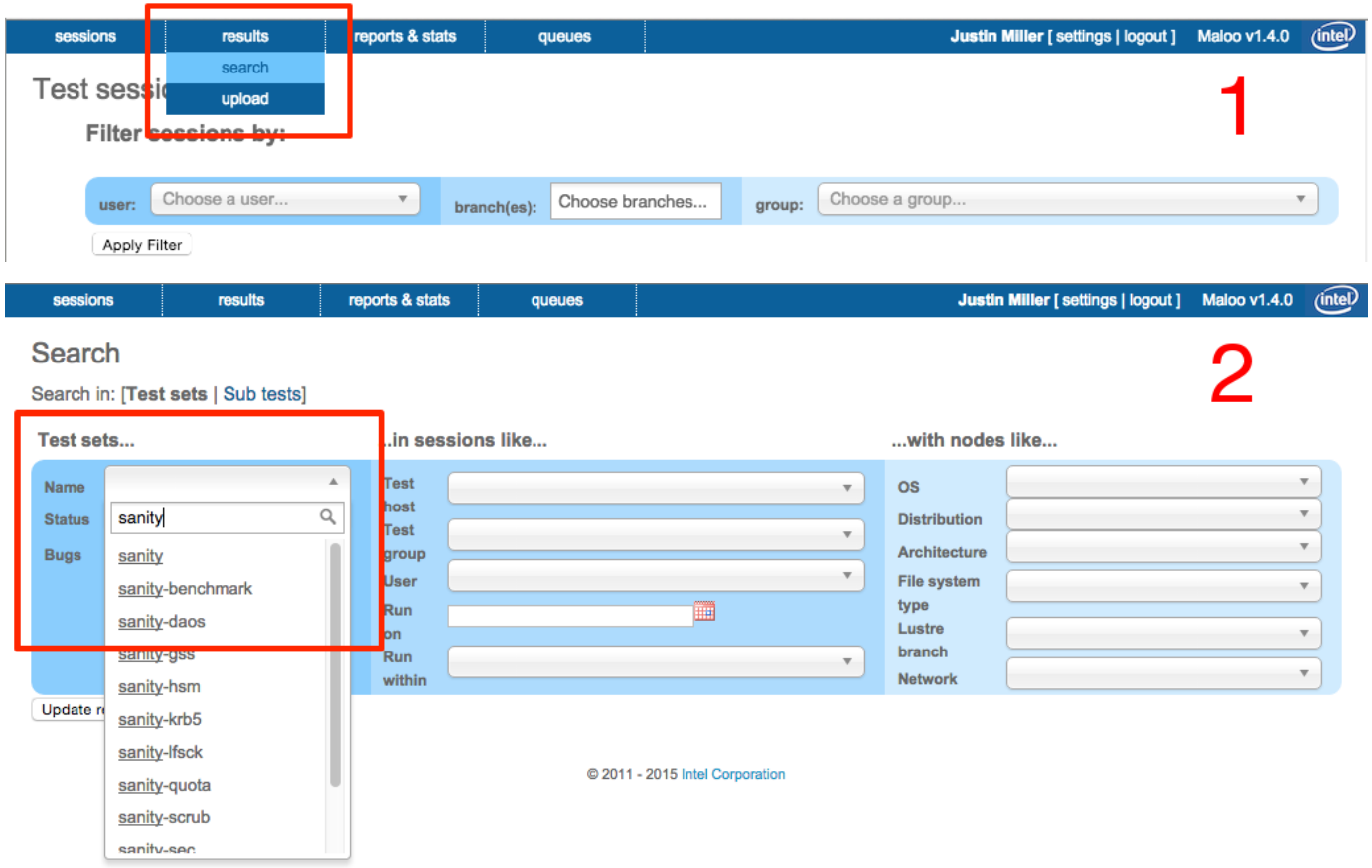
Reporting Test Failures

Community Release Results Database

- Look for known failures
- Ability to upload your results

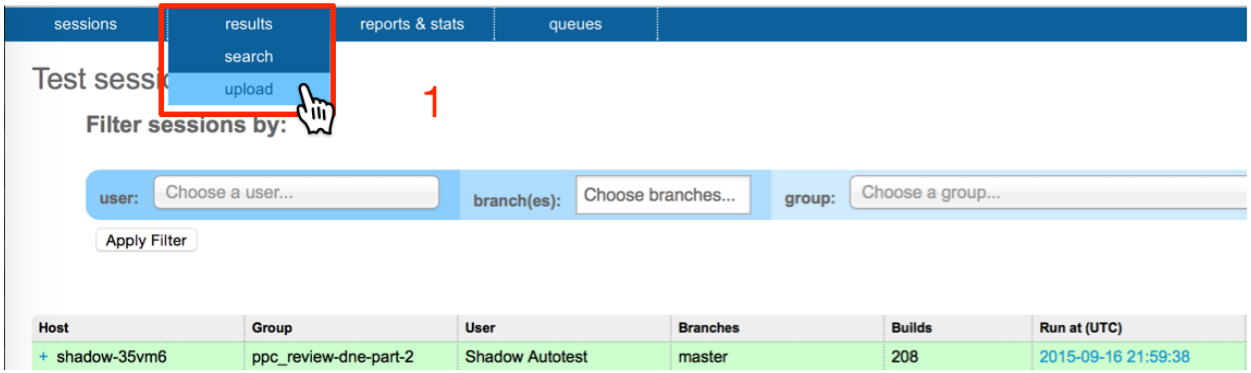
Community Release Issue Tracker

- Search for failing test or resolution



Search the Community Release Results DB for a specific test

http://testing.hpdd.intel.com



Upload Completed Results to the Community Release Results DB
http://testing.hpdd.intel.com

Reporting a Bug

Community Release Issue Tracker Ticket

- Operating System and Linux kernel version
- Lustre version on client and servers
- Details on hardware configuration, including ldiskfs or zfs
- MDT/OST type
- Recommended - Upload LTS log files

Call to Action

Administrators

- Run ‘regression’ test group on your systems

Developers

- “Test early, test often”
- Add tests for features lacking coverage

Everyone

- Run all of the actively maintained tests
- Report any known reproducers, inside or outside of LTS

References

Rutman LAD ‘12:

http://wiki.opensfs.org/images/b/b1/Test_Framework_2012.pdf

General Testing How to:

http://wiki.lustre.org/Testing_HOWTO
https://wiki.hpdd.intel.com/display/PUB/Testing+a+Lustre+filesystem

Environment Variables

https://wiki.hpdd.intel.com/display/PUB/Lustre+Test+Tools+Environment+Variables

Community Release Results Database

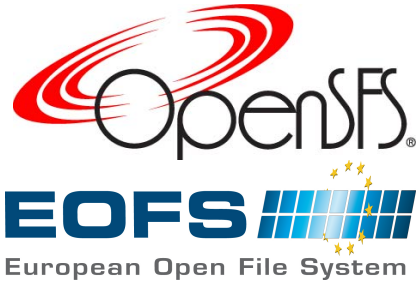
https://testing.hpdd.intel.com/test_sessions

Community Release Results Database

https://jira.hpdd.intel.com/secure/Dashboard.jspa

Grigoryev LAD ‘14:

http://www.eofs.eu/fileadmin/lad2014/slides/08_Roman_Grigoryev_Xperior__LAD14_Seagate.pdf



Thank you.
Questions?

Justin Miller - jmiller@cray.com
James Nunez - james.a.nunez@intel.com


intel
Software


CRAY
THE SUPERCOMPUTER COMPANY


EOFS
European Open File System
OpenSFS