# Lustre 1.8 Release Test Plan

| Author | Date | Description of Document Change | Client Approval By | Client Approval Date |
|---|---|---|---|---|
| Chameleon Team | 06/13/08 | First draft. | | |
| Jian Yu | 06/30/08 | Second draft. Add the Lustre 1.8 build schedule and improve Test Cases section. | | |
| Jian Yu | 07/04/08 | Third draft. Improve the Lustre 1.8 Build 01 test matrix. Add Build 02 test matrix. | | |
| Jian Yu | 07/14/08 | Fourth draft. Update the test plan to make it only conduct the Lustre 1.8 release candidate testing. | | |
| Jian Yu | 08/11/08 | Update the RC1 test matrix. | | |
| Yibin Wang | 11/20/08 | Add interop testing with b1_6 and HEAD | | |
| Yibin Wang | 11/28/08 | Add CentOS5 to test matrix. Use YALA as much as possible. | | |

# I. Test Plan Overview

This test plan is intended to conduct the release testing of Lustre 1.8. The goal is to find and resolve defects in the Lustre codes during the release testing cycles. Additional testing will be added for new features and will be documented in the test plans for those features.

Collaboration with the Open Source community.

## Executive Summary

The Lustre QE team will require a test plan that can verify the Lustre 1.8 release.

- Create a 2-week test matrix to verify each release candidate (RC) for Lustre 1.8.
- Inputs from QE and the Release Management Group (RMG) team will be required.
- All QE test nodes will be used in this testing.
- The output will be a state of the release candidate when completed.

## Problem Statement

Before a new Lustre version can be released, the Lustre codes for that version must pass a release testing cycle. During the whole testing cycle, there used to be several release candidates made because new regressions were found and fixed after each previous release candidate testing. This increased the time and effort required to get the new Lustre version released. A good release test plan can conduct the testing to find regressions earlier, more widely and deeply so as to shorten the time for making the new Lustre version production ready.

## Goal

The goal is to find and resolve defects in the Lustre codes during the release testing cycle and drive the Lustre codes to be production ready and ready for release.

## Success Factors

A better QE Lustre test plan that can be distributed to the Open Source community will give visibility into our test processes and provide a mechanism for the Open Source community to assist with our testing efforts.

## Testing Plan

| **Define the setup steps that need to happen for the hardware to be ready? Who is responsible for these tests?** |
|---|
| In manual testing, the testing environment setup steps are as follows:<br>1) Reserve test cluster time through Cluster Scheduler<br>2) Set up the test cluster by running OSLO on lts-head node<br>3) Configure Lustre file system and start running the tests<br>4) Send test results to Buffalo by running send_report.pl on lts-head node.<br><br>QE team in the Lustre group is responsible for setting up the test environment, running the tests, vetting and reporting the test results.<br><br>In automated testing, the testing environment setups are as follows:<br>1) Submit YALA testing request in YALA<br>2) Vet the test result on Buffalo.<br><br>The release manager is responsible for step 1; QE is responsible for step 2. |

| **Specify the date these tests will start, and length of time that these test will take to complete.** |
|---|
| Each series of tests will be run when a new RC has been created. |
| The proposed length of time is a 2-week test effort for one RC. |

| **Specify (at a high level) what tests will be completed?** | |
|---|---|
| Functional Tests | acceptance small test suite, Cascading_rw, Connectathon |
| Performance Tests | IOR, PIOS, Metabench, Compilebench, LST |
| Quotas Tests | |
| Recovery Tests | |
| Interoperability Tests | B1_8 interop with b1_6 and HEAD |
| Stress Tests | Simul, Racer |
| Upgrade/Downgrade Tests | |

## Test Cases

| Functional Test Cases | | |
|---|---|---|
| All of the test cases in the acceptance small test suite: | | |
| No. | Test Case | Description |
| 1. | RUNTESTS | Basic regression tests with unmounting/remounting. |
| 2. | SANITY | Tests that verify operation under normal operating conditions. |
| 3. | DBENCH | dbench benchmark for simulating N clients to produce the file system load. |
| 4. | BONNIE | Bonnie++ benchmark for creation, reading, and deleting many small files. |
| 5. | IOZONE | Iozone benchmark for generating and measuring a variety of file operations. |
| 6. | FSX | File system exerciser. |
| 7. | SANITYN | Tests that verify operations from two clients under normal operating conditions. |
| 8. | LFSCK | Tests e2fsck and lfsck to detect and fix file system corruption. |
| 9. | LIBLUSTRE | Runs a test linked to a liblustre client library. |
| 10. | REPLAY_SINGLE | Tests that verify recovery after MDS failure. |
| 11. | CONF_SANITY | Tests that verify configuration. |
| 12. | RECOVERY_SMALL | Tests that verify RPC replay after communications failure. |
| 13. | REPLAY_OST_SINGLE | Tests that verify recovery after OST failure. |
| 14. | REPLAY_DUAL | Tests that verify recovery from two clients after server failure. |
| 15. | INSANITY | Tests multiple concurrent failure conditions. |
| 16. | SANITY_QUOTA | Tests that verify filesystem quotas. |
| 17. | REPLAY_VBR | Tests that verify version based recovery features. |
| 18. | PERFORMANCE_SANITY | Performance tests ported from CMD3 test suites. |
| Other functional test cases: | | |
| 19. | Cascading_rw | An MPI coordinated test of parallel cascading read/write. |
| 20. | NFS Connectathon | An industry standard to verify basic NFS functionality. |
| 21. | statahead | Statahead performance test. |
| 22. | write_append_truncate | Append and truncate write test (MPI). |
| 23. | write_disjoint | Disjoint write test. |

| Performance Test Cases | | |
|---|---|---|
| No. | Test Case | Description |
| 1. | IOR | Industry HPC IO performance benchmark for testing performance of parallel file systems using various interfaces and access patterns. |
| 2. | PIOS | Lustre IO performance benchmark for evaluating backend devices and file systems. |
| 3. | Metabench | Industry metadata performance benchmark for parallel file system. |
| 4. | Compilebench | Industry benchmark, which tries to age a filesystem by simulating some of the disk IO common in creating, compiling, patching, stating and reading kernel trees. |
| 5. | LST | LNET self-test, which is used for benchmarking Lustre network performance. |

| Quotas Test Cases |
|---|
| CVS path: doc/TP/quotas-tp.odt |
| Extra quota testing could be run along with the upgrade testing:<br>1) Before the upgrade, three users would be setup with different limit of quotas.<br>The following operations would be run separately to consume the users' quota spaces:<br>a) IOR -w -k<br>b) extract kernel tarball<br>c) iozone -i 0 -+d -w<br>iozone operation would exceed the quota limit for the corresponding quota user.<br>After the above operations are finished, the final state of quota usage/limits for the three users would be recorded.<br>2) Upgrade the MDS and OSSs. Before and after upgrading the Lustre clients, record the quota usage/limits for the three users and verify that they are the same as before the upgrade.<br>3) Verify the file data created by the above operations are not affected by the upgrade.<br>4) Verify quota user that ran out of quotas is still out of quotas after the upgrade. |

| Recovery Test Cases |
|---|
| CVS path: doc/TP/VBR_tp.odt and doc/TP/VBR_phase2_tp.odt |

| Interoperability Test Cases |
|---|
| CVS path: doc/TP/interoperability-tp.odt |

| Stress Test Cases | | |
|---|---|---|
| No. | Test Case | Description |
| 1. | Simul | An MPI coordinated test of parallel filesystem system calls and library functions. It was designed to perform filesystem operations simultaneously from many nodes and processes to test the correctness and coherence of parallel filesystems. |
| 2. | Racer | Test for file system race conditions by concurrently creating, moving, deleting, etc. a set of files. |
| 3. | low-memory | Tests would be run on the Lustre test cluster setting up with low memory (512M). |
| 4. | multi-client-per-node | Multiple Lustre clients mounted on each Lustre client node while running IOR/Iozone/Simul. |
| | | Five or more Lustre clients would be mounted on each of the client nodes, and the following tests would be run:<br>1) Use pdsh to run iozone on all of the Lustre clients on all of the client nodes concurrently<br>2) Use mpirun to have a single IOR running across the same mount point on all of the client nodes (totally 5 IORs will be run concurrently) Running in both file-per-process (for basic functionality testing) and single-shared-file (for actual interop testing) modes.<br>3) Use mpirun to have a single Simul running across the same mount point on all of the client nodes (totally 5 Simuls will be run concurrently) |

| Upgrade/Downgrade Test Cases |
|---|
| Upgrade/downgrade testing could be run on a Lustre cluster with configuration covering different upgrade/downgrade paths separately on client(s), MDS and OSS(s).<br><br>The following paths should be tested:<br>1) Upgrade from the latest 1.4.x release to the latest 1.6.x release, then to the latest 1.8 RC.<br>2) Downgrade from the latest 1.8 RC to the latest 1.6.x release.<br><br>The upgrade/downgrade operations need be performed in the following two ways:<br>1) Rolling upgrade/downgrade<br>Individual Lustre servers (or their failover partners) and clients are upgraded/downgraded one at a time. Some applications (iozone, tar, etc.) would be run on the live clients while failover upgrading/downgrading the servers. No application failure should occur.<br>2) Clean upgrade/downgrade<br>Shut down the entire filesystem and upgrade/downgrade all servers and clients at once.<br><br>NOTE: The upgrade/downgrade testing between Lustre 1.8 and Lustre 2.0 will be run during the Lustre 2.0 release testing. |

# III. Supported Architectures

| Architectures |
|---|
| x86_64 |
| i686 |
| ia64 |
| ppc64 (only on client) |

## Supported Distributions and Kernels

| Distribution | Kernel |
|---|---|
| SLES 10 | 2.6.16.60-0.31 |
| RHEL 5/CentOS 5 | 2.6.18-92.1.10.el5 |
| SLES 9 | 2.6.5-7.314 |
| RHEL 4 | 2.6.9-67.0.22.EL |
| | vanilla 2.6.22.14 |

NOTE: The above-supported kernels for different distributions are for Lustre 1.8.0 RC1.

## Platforms, Network Types, Client Types for Testing

| | RHEL 4 | RHEL 5 | CentOS 5 | SLES 9 | SLES 10 | vanilla 2.6.22 |
|---|---|---|---|---|---|---|
| x86_64 | √ | √ | √ | √ | √ | √ |
| i686 | √ | √ | √ | √ | √ | |
| ia64 | √ | √ | | | | |
| ppc64 (only on client) | | | | √ | √ | |
| TCP (1GigE) | √ | √ | √ | √ | √ | √ |
| IB (OFED 1.3.1) | √ | √ | | | √ | |
| Patched client | √ | | √ | √ | | |
| Patchless client | | √ | | | √ | √ |

NOTE:
The above matrix shows the platforms, network types, etc. for Lustre 1.8 release testing, which does not mean that Lustre 1.8 only supports the above-listed network types.

## Proposed Time Line

With a 2-week test effort on one Lustre 1.8 RC, QE will not be able to run all tests on all supported architectures/distributions. Test matrices for different RCs would be created to show different test coverage. A full test coverage would be performed over the complete release testing cycle.

The automation test system (YALA) can be used to the maximum extent possible to do automated acc-sm testing, so that QE could be freed from running acc-sm tests and only need spend time vetting the test results.

## Scale Testing

Limited scale testing is run by the Lustre QE team. At scale testing by our partners and the Open Source community will ensure a more robust and quality Lustre product.

## Automation Testing

Automation will be used by SUN QE to support this time line. The automation is not for external use. Acceptance small test suite is available in the Lustre source tree and can be used by the Open Source community.

## Not Tested

QE will not test all Lustre configurations (i.e., 1Cx1Mx1O, 1Cx1Mx2O, 1Cx1Mx3O...).
QE will not test all type of NIC and HBA including its drivers.
QE will not test all machine configuration (i.e., number of CPU, memory size, storage capacity...).
QE will not cover all industry performance tests.

# II. Test Schedule

## Lustre 1.8.0

Recommended e2fsprogs version: 1.40.11-sun1

Tracking tickets on Bugzilla:

- Lustre 1.8.0 release tracker: Bug 12662
- Lustre 1.8.0 release testing tracker: TBD

### Lustre 1.8.0 RC1

1. CVS tag: v1_8_0_RC1
2. Time Plan:
   - *Start date:* TBD
   - End date: TBD (should be 2 weeks after 'Start date')
3. Test Matrix(b1_8 normal release testing)

| ***Run by*** | | Jack | Yep | Wangyb | Wangyb | Yep | Jack | Yep |
|---|---|---|---|---|---|---|---|---|
| | **Platform** / **Test case** | SLES10/ x86_64 | SLES10/ i686 | RHEL5/ x86_64 | RHEL5/ia64 +RHEL5/i686 (C+S) | SLES9/ppc64 +RHEL4/x86_64 (C+S) | SLES10 (vanilla 2.6.22.14)/x86_64 | CentOS5 /x86_64+ SLES9/i686(C+S) |
| YALA | acc-sm | [0][2][7][8] | [2][3][5] | [2][10] | [0][1][2][4][7] | [1][7][8] | [2] | [0] |
| YALA | IOR | [2][3][11] | | | | | | |
| YALA | PIOS | [2][3][12] | | | | | | |
| YALA | Metabench | [2][3][12] | | | | | | |
| YALA | Compilebench | [2][3][12] | | | | | | |
| YALA | LST | [2][3] | | | | | | |
| YALA | Racer | | [2][6][7] | | | | | |
| YALA | Simul | | | [2][6][7] | | | | |
| YALA | Cascading_rw | | | | | [1][6][7] | [2][6][7] | |
| YALA | Connectathon | | | | [1][2][9] | | | |
| QE | *Statahead* | | | [14] | | | | |
| QE | *write_append_truncate* | | | | | [17] | | |
| QE | *write_disjoint* | | | | [18] | | | |
| Wangyb | Interop | [13] | [13] | [13] | [13] | [13] | [13] | |
| Max S. | LOL upgrade | | | | | | | |

**NOTE:**

[0] – Run test manually because YALA does not support PPC64 and IA64 and CentOS5 yet.
[1] - Run test with intermixed distros/archs on a Lustre client and servers.
[2] - Run test on patchless client.
[3] - Run test over IB (OFED 1.3.1) network.
[4] - Run test under low memory (512M).
[5] - Run acc-sm tests including performance-sanity test suite.
[6] - Run test with quotas on (setting quota limits to be just large enough for the test to complete).
[7] - Run test with flock locking enabled (mounting Lustre client(s) with "flock" option).
[8] - Run test on a ppc64 Lustre client which connects to x86_64 Lustre servers.
[9] - Mount Lustre client with "localflock" option.
[10] - Run the following recovery tests with HARD failure mode on two clients (see VBR Test Plan):

1) replay-vbr
2) insanity
3) replay-single
4) recovery-small
5) replay-dual
6) conf-sanity test 27b

[11] - Run performance test with file system quotas off and on.
[12] - Run performance test covering the following scenarios:

1) Native Lustre
2) NFSv3 over Lustre
3) NFSv4 over Lustre

[13] - Interoperability, upgrade, downgrade, multi-client-per-node, extra quota testing:

Intermix different platforms and Lustre versions as follows:

Clients:

C1: SLES10 vanilla 2.6.22.14/x86_64/patchless v1_8_0_RC1
C2: SLES10/i686/upgrade from v1_4_12_RC6 to patchless v1_6_5_1_RC2, then to patchless v1_8_0_RC1
C3: RHEL5/i686/patchless v1_6_5_1_RC2
C4: RHEL5/x86_64/patchless v1_8_0_RC1
C5: SLES9/ppc64/v1_8_0_RC1
C6: RHEL5/ia64/upgrade from patchless v1_6_5_1_RC2 to patchless v1_8_0_RC1
C7: RHEL4/ia64/v1_4_12_RC6

MDS:

M1: RHEL4/x86_64/upgrade from v1_4_12_RC6 to v1_6_5_1_RC2, then to v1_8_0_RC1, then downgrade to v1_6_5_1_RC2

OSSs:

O1: SLES10/x86_64/upgrade from v1_4_12_RC6 to v1_6_5_1_RC2, then to v1_8_0_RC1

Rolling upgrade/downgrade will be performed. Iozone and tar would be run on the live clients

while failover upgrading/downgrading the servers. No application failure should occur.

Before/after the upgrade, extra quota testing would be performed (see Quotas Test Cases).

After finishing the upgrade/downgrade testing, multi-client-per-node testing would be run (see Stress Test Cases).

[14] - test statahead to verify *15927*

mount 5 clients on a node, run multiple(5) simultaneous kernel tar/untar/diff processes.

[15] – For acc-sm testing on YALA, we need to check 'Run acc-sm with Option SLOW=yes" checkbox.

[16] – upgrade LOL to v1_8_0_RC1. Make sure that there is no bug during/after the upgrade.

[17] – steps to do it

a. mount lustre on multiple nodes in /mnt/lutre;

b. mkdir /mnt/lustre/wat;

c. put the nodes names in mpinodes, let np=(number of nodes+4) so that we have 4 MPI procs run on each node;

d. mpirun -np $np -machinefile mpinodes /usr/lib*/lustre/tests/write_append_truncate /mnt/lustre/wat/wat.file 10000 (you can increase test time by increase this number.)

[18] – steps to do it:

a. mount lustre on multiple nodes in /mnt/lutre;

b. mkdir /mnt/lustre/wd;

c. put the nodes names in mpinodes, let np=(number of nodes+4) so that we have 4 mpi procs run on each node;

d. mpirun -np 4 -machinefile mpinodes /usr/lib*/lustre/tests/write_disjoint -f /mnt/lustre/wd/wd.file [-n 10000]. (you can use the '-n 10000' to increase test load.)

4. Test Matrix(b1_8 interop test with b1_6&HEAD)

| QE / Test | Jack | Yep | Wangyb |
|---|---|---|---|
| MDS | centos5/i686[0] | sles10/i686[1] | rhel4/ia64[2] |
| OST | rhel4/x86_64[1] | rhel5/x86_64[0] | Vanilla-2.6.22/x86_64[1] |
| CLIENT | sles10/ppc64[2] | sles9/i686[2] | sles10/i686[0] |
| acc-sm | TEST[5][10] | TEST[9] | TEST[4] |
| IOR | TEST[5][10] | | |
| PIOS | TEST[5][10] | | |
| Metabench | TEST[5][10] | | |
| Comilebench | TEST[5][10] | | |
| LST | TEST[5][10] | | |
| Simul | | | TEST[4][7] |
| Connectathon | | TEST[9] | |
| Racer | | TEST[9] | |
| Cascading_rw | | | TEST[4][7] |

[0]Branch=b1_8
[1]Branch=b1_6
[2]Branch=HEAD
[3]Run test by YALA
[4]Run test on patchless client.
[5]Run test over IB (OFED 1.3.1) network.
[6]Run test under low memory (512M)
[7]Run test with quotas on (setting quota limits to be just large enough for the test to complete)
[8]Run test with flock locking enabled (mounting Lustre client(s) with "flock" option)
[9]Run test with flock locking enabled (mounting Lustre client(s) with "localflock" option).
[10]Run test on a ppc64 Lustre client

**Test matrix for Lustre 1.8.0 RC2 would be created right before the RC2 is going to be tagged.**

# III. Test Plan Approval

- Review date for the Test Plan review with the RMG
  - 06/13/2008 – reviewed by Andreas and Mallik
  - 07/03/2008 – reviewed by Andreas and Robert Read
- Date the Test Plan was approved by the RMG
- Date(s) agreed to by the RMG to conduct testing