Demonstration Milestone Completion for the Striped Directories subproject of the Distributed Namespace Project of the SFS-DEV-001 contract.

Revision History

Date	Revision	Author
2015-07-01	Draft	R. Henwood
2015-08-06	1.0	J. Popp

Table of Contents

Introduction

The following milestone completion document applies to Subproject 2.2 – Striped Directories (also known as DNE2) of the Lustre[1] Distributed Namespace Project of the OpenSFS Lustre Development contract SFS-DEV-001 signed July 30th, 2011. The specific items executed for the Demonstration milestone and listed below were approved on 19th May 2015.

The DNE2 code is functionally complete and recorded in the Implementation Milestone. The purpose of this Milestone is to verify the code performs acceptably in a production-like environment. In addition to completing all the Test Scenarios (demonstrated for the Implementation Milestone,) DNE2 Performance has been measured as recorded below.

The tests were executed on a variety of hardware platforms include the OpenSFS Functional Test Cluster and the public cloud. Details of the hardware are available in Appendix A. For all the tests, Lustre software Master with DNE2 patches was used.

Subproject Description

Per the scope statement, the project is described as follows:

> Today, Lustre filesystem routinely have thousands to tens of thousands of clients. As client numbers continue to increase, a single Metadata Server (MDS) for a single Lustre filesystem becomes a performance and scalability constraint. This restraint has been partially lifted with the completion of the DNE1: Remote Directories sub-project which spread sub-directories on Lustre onto independent MDS nodes and MDTs. A limitation still exists, however, for individual directories. Currently a single directory can only be a single MDT served by a single MDS. This restriction limits both the quantity and performance of files in any given directory.

Milestone Completion Criteria

Per the contract, Implementation milestone is described as follows:

Demonstration. *Upon functional completion of the feature, Contractor shall demonstrate the appropriate functionality of the project. This shall be done through execution of test cases designed to prove the acceptance criteria defined during the Solution Architecture.*

Demonstration specifics will be defined and mutually agreed to for each subproject in the scope and architecture phases.

---

1 Other names and brands may be the property of others.

- ***Functional Test Plan****: Contractor shall develop and recommend a functional test plan, as defined by OpenSFS, designed to demonstrate the functional completeness of the feature. The results of functional testing with supporting documentation will be presented to OpenSFS for review.*
- ***Performance Test Execution****: Contractor shall define and recommend a set of performance tests as defined by OpenSFS to document the performance characteristics for performance related features. Contractor shall execute these tests and present results of these tests to OpenSFS for review. OpenSFS shall provide adequate test platforms when scale is necessary for performance testing as recommended by Contractor and defined by OpenSFS.*

Functional Test

Basic Functional Regression Testing

As described in the Solution Architecture, the basic Lustre regression tests have been updated to run DNE-specific functionality. The basic testing will be run on a regular basis for all Lustre patches landed to the `master` branch to ensure DNE striped directory functionality continues to work in the future. These tests are designed to verify specific basic functionality in an efficient manner and are intended to run in the minimum time possible so that they can be run on all patches.

Status: **Complete**

http://review.whamcloud.com/15163

Striped Directory Creation

```
usage: setdirstripe <--count|-c stripe_count> [--index|-i mdt_index]
   [--hash-type|-t hash_type] [--default_stripe|-D ] <dir>
 stripe_count: stripe count of the striped directory
 mdt_index: MDT index of first stripe
 hash_type: hash type of the striped directory
 default_stripe: set default striping parameters of the directory
```

Regression Tests

The test script creates all top-level test subdirectories on remote MDTs. Wherever possible, these directories are automatically striped across multiple MDTs using the `test_mkdir()` helper function. This ensures that a majority of the regression tests run as part of these scripts are also exercised with DNE1 and DNE2 functionality. Additional DNE2 specific functionality tests have been added to these scripts. Unless otherwise specified, all operations shall succeed without error. The list of tests to sanity.sh to specifically exercise DNE2 functionality include:

sanity.sh: Tests to verify operation under normal operating conditions

- **test_17n** "run e2fsck against master/slave MDT which contains remote dir"

  For ldiskfs-backed MDTs create remote striped directories and create files within those directories, then unmount the MDTs and verify with e2fsck that the on-disk structure of each MDT is consistent. Remount the MDTs and remove the created files and directories and again verify with e2fsck that the on-disk structure of each MDT is consistent. Create a directory on MDT0 and migrate it to another MDT and then verify with e2fsck that the on-disk structure is consistent.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/81f396b2-0e8d-11e5-828a-5254006e85c2

- **test_24x** "cross MDT rename/link"

Create local and remote striped subdirectories, and a regular file within each. Rename the remote directory to the local parent directory. Rename the file in the local directory over the file in the remote directory. Create a local file and hard link it into the remote directory.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/8318f942-0e8d-11e5-828a-5254006e85c2

- **test_24y** "rename/link on the same dir should succeed"

Create a remote striped subdirectory, and a pair of regular files and directories within it. Rename one directory over the second directory. Rename one file over the second file. Link a new name to the existing file.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/83238a92-0e8d-11e5-828a-5254006e85c2

- **test_24E** "cross MDT rename/link"

Create two remote striped subdirectories on different MDTs, and a pair of regular files and directories within it. Rename one directory over the second directory. Rename one file over the second file. Link a new name to the existing file.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/83514180-0e8d-11e5-828a-5254006e85c2

- **test_31p** "remove of open striped directory"

Create a striped directory and set a default directory striping pattern on it. Create subdirectories therein, open the directories, and unlink them. Verify that the unlinked subdirectories are not accessible to the namespace.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/85cd5fc0-0e8d-11e5-828a-5254006e85c2

- **test_33d** "openfile with 444 modes and malformed flags under remote dir"

Create a remote subdirectory, and create a read-only file within it that is owned by a non-root user. Open file as the non-root user in read-write mode and verify that the open fails. Open/create a new read-only file as a regular user in read-write mode should succeed. Opening the file again in read-write mode should fail. Open the file with invalid flags should fail.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/867a0a9a-0e8d-11e5-828a-5254006e85c2

- **test_33f** "nonroot user can create, access, and remove a striped directory"

Enable `mdt.*.enable_remote_dir_gid=-1` on the MDS nodes. Create a remote striped directory as a non-root user. As the same user, create files within that directory, remove them, then remove the directory.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/86889a88-0e8d-11e5-828a-5254006e85c2

- **test_39p** "remote directory cached attributes updated after create"

Create two remote striped subdirectories in a local parent directory. Verify the nlink count on the parent directory is correct. Remove one remote subdirectory and verify the nlink count is decremented.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/87eb3e80-0e8d-11e5-828a-5254006e85c2

- **test_154b** "Open-by-FID for remote directory"

Create a remote directory and create a regular file on the remote MDT. Verify that the `$MOUNT/.lustre/fid/` special directory works to open the file on the remote MDT. Verify that `lfs fid2path` works on the remote file.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/8f4bdee6-0e8d-11e5-828a-5254006e85c2

- **test_161b** "link ea sanity under remote directory"

Create a remote directory, and two subdirectories and a regular file on the remote MDT. Create hard links from the regular file to the remote MDTs. Verify that `lfs fid2path` lists the paths to all names from all MDTs. Rename the file to another MDT and verify that `lfs fid2path` shows the correct new filename.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/8fff2938-0e8d-11e5-828a-5254006e85c2

- **test_230a** "Create remote directory and files under the remote directory"

Create a remote striped directory and two remote subdirectories, and a number of regular files on the remote MDT. Verify that the directory and files are indeed created on the expected MDT, the same as the parent directory.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/9464f282-0e8d-11e5-828a-5254006e85c2

- **test_230b** "migrate directory"

Create a remote directory and a number of local and remote subdirectories and regular files with hard links and containing verifiable. Migrate the directory tree using `lfs mv` and verify no error was expected. Verify the original file contents have not changed.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/946c315a-0e8d-11e5-828a-5254006e85c2

- **test_230c** "check directory accessibility if migration is failed"

Create a remote directory and local and remote subdirectories, along with regular files containing verifiable data. Start to migrate the directory tree using `lfs mv`, but interrupt the migration before it completes. Verify that the interrupted migration directory is still accessible after the MDT is available again.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/94757ab2-0e8d-11e5-828a-5254006e85c2

- **test_230d** "check migrate big directory"

Create a remote directory and a larger number of subdirectories and files therein (10000). Migrate the parent directory to a new MDT and verify the file is on the target MDT and ensure the file data is correct.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/0daf5c50-143c-11e5-9804-5254006e85c2

- **test_300g** "check default striped directory for normal directory"

Create a local directory and set a variety of different default directory striping patterns on it. For each pattern, create subdirectories and verify that they properly inherit the default directory striping pattern. Remove the subdirectories. Remove the default directory striping pattern.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/96ddb10c-0e8d-11e5-828a-5254006e85c2

sanityn.sh: Tests to verify operations from two clients under normal operating conditions

- **test_81** "rename and stat under striped directory"

Create a local directory, and create striped subdirectories beneath it. Create a regular file beneath the striped directory and verify that they can be renamed between directories. Verify that the renamed file is not visible on the first mountpoint. Rename the file on the second mountpoint back to the original filename, and verify it is again visible on the original mountpoint.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/12516edc-0e8e-11e5-828a-5254006e85c2

conf-sanity.sh: Tests to verify configuration is working correctly.

- **test_32c** "dne upgrade test"

  Upgrade 2.4 and 2.5 pre-populated filesystem images to have multiple MDTs. Verify that the pre-existing file data is accessible, and that new operations on the upgraded filesystems work correctly. Verify that common configuration operations are working correctly on all MDTs.

  Status: **Passed**

  https://testing.hpdd.intel.com/sub_tests/362c3c10-0e8e-11e5-828a-5254006e85c2

recovery-small.sh: Tests to verify RPC replay after communications failure (message loss)

- **test_110a** "create remote directory: drop client req"
- **test_110b** "create remote directory: drop master reply to client"
- **test_110c** "create remote directory: drop update rep on slave MDT"
- **test_110d** "remove remote directory: drop client req"
- **test_110e** "remove remote directory: drop master reply to client"
- **test_110f** "remove remote directory: drop master reply to client"
- **test_110g** "drop master reply during migration"
- **test_110h** "Cross-MDT file rename: drop slave MDT update reply"
- **test_110i** "Cross-MDT dir rename: drop slave MDT update reply"
- **test_110j** "Cross-MDT file link: drop slave MDT update reply"

  Status: **Passed**

  https://testing.hpdd.intel.com/test_sets/50504b5e-0e8e-11e5-828a-5254006e85c2

replay-single.sh: Tests to verify recovery after MDS failure

- **test_80a** "create remote dir, drop update rep from MDT0, fail MDT0"
- **test_80b** "create remote dir, drop update rep from MDT0, fail MDT1"
- **test_80c** "create remote dir, drop update rep from MDT1, fail MDT0, then MDT1"
- **test_80d** "create remote dir, drop update rep from MDT1, fail 2 MDTs at the same time"
- **test_80e** "create remote dir, drop MDT1 rep, fail MDT0"
- **test_80f** "create remote dir, drop MDT1 rep, fail MDT1"
- **test_80g** "create remote dir, drop MDT1 rep, fail MDT0, then MDT1"
- **test_80h** "create remote dir, drop MDT1 rep, fail 2 MDTs at the same time"
- **test_81a** "unlink remote dir, drop MDT0 update rep, fail MDT1"
- **test_81b** "unlink remote dir, drop MDT0 update reply, fail MDT0"
- **test_81c** "unlink remote dir, drop MDT0 update reply, fail MDT0, then MDT1"
- **test_81d** "unlink remote dir, drop MDT0 update reply, fail 2 MDTs"
- **test_81e** "unlink remote dir, drop MDT1 req reply, fail MDT0"
- **test_81f** "unlink remote dir, drop MDT1 req reply, fail MDT1"
- **test_81g** "unlink remote dir, drop req reply, fail MDT0, then MDT1"
- **test_81h** "unlink remote dir, drop request reply, fail 2 MDTs"

- **test_100a** "create striped dir (master stripe is on MDT0), drop update rep from MDT1, fail MDT1"
- **test_100b** "create striped dir(master stripe is on MDT0), fail MDT0"
- **test_110a** "create striped dir(master stripe is on MDT1), fail MDT0"
- **test_110b** "create striped dir(master stripe is on MDT1), fail MDT0 and client"
- **test_110c** "create striped dir(master stripe is on MDT1), fail MDT1"
- **test_110d** "create striped dir(master stripe is on MDT1), fail MDT1 and client"
- **test_110e** "create striped dir(master stripe is on MDT1), uncommit on MDT1, fail client & MDT0 & MDT1"
- **test_110f** "create striped dir(master stripe is on MDT1), fail MDT0 & MDT1"
- **test_110g** "create striped dir(master stripe is on MDT1), uncommit on MDT0, fail client & MDT0 & MDT1"
- **test_111a** "unlink striped dir(master stripe is on MDT1), fail MDT0"
- **test_111b** "unlink striped dir(master stripe is on MDT1), fail MDT1"
- **test_111c** "unlink striped dir(master stripe is on MDT1), uncommit on MDT0, fail client & MDT0 & MDT1"
- **test_111d** "unlink striped dir(master stripe is on MDT1), uncommit on MDT1, fail client & MDT0 & MDT1"
- **test_111e** "unlink striped dir(master stripe is on MDT1), uncommit on MDT1, fail MDT0 & MDT1"
- **test_111f** "unlink striped dir(master stripe is on MDT1), uncommit on MDT0, fail MDT0 & MDT1"
- **test_111g** "unlink striped dir(master stripe is on MDT1), fail MDT0 & MDT1"
- **test_112a** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0"
- **test_112b** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT1"
- **test_112c** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT2"
- **test_112d** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT3"
- **test_112e** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT1"
- **test_112f** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT2"
- **test_112g** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT3"
- **test_112h** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT1 & MDT2"
- **test_112i** "cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT1 & MDT3"
- **test_112j** "Cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT2 & MDT3"

- **test_112k** "Cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT1 & MDT2"
- **test_112l** "Cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT1 & MDT3"
- **test_112m** "Cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT0 & MDT2 & MDT3"
- t**est_112n** "Cross MDT rename, (src_dir is on MDT0, src_child is on MDT1, tgt_dir is on MDT2, tgt_child is on MDT3), fail MDT1 & MDT2 & MDT3"

Status: **Passed**

https://testing.hpdd.intel.com/test_sets/296cdcd0-0ea9-11e5-bae1-5254006e85c2

Upgrade test

1. Create a filesystem with single MDT with Lustre 2.5.
2. Create a directory tree and populate it with 1M regular files
3. Upgrade the filesystem to 2.7 + DNE async commit patches.
4. Add 3 new MDTs to the filesystem.
5. Create a striped directory with stripe_count = 4.

```
lfs setdirstripe -c4 /mnt/lustre/striped_dir
```

6. Create 1M files under the striped directory
7. Verify that both the original 1M files exist
8. Verify that the new 1M files are approximately evenly distributed across all four MDTs
9. Unlink both sets of files
10. Unlink the striped directory
11. No errors will be observed

Status: **Passed**

Test was completed on the OpenSFS test cluster on 2nd June 2015 and again on 3rd August 2015. The latest test results are attached to https://jira.hpdd.intel.com/browse/LU-6946

Default stripe count test

1. Create a filesystem with 4 MDTs, 4 OSTs, and 4 clients
2. Then create a testdir and set default stripe count = 2

```
mkdir /mnt/lustre/test
lfs setdirstripe -D -c2 /mnt/lustre/test
 # all of its children will be striped directory with stripe_count = 2.
```

3. Run mdtest to validate `/mnt/lustre/test`

```
mdtest -n 100 -i 3 -d /mnt/lustre/test
```

4. No errors will be observed, and correct striping will be observed.

Status: **Passed**

https://testing.hpdd.intel.com/sub_tests/96ebdfca-0e8d-11e5-828a-5254006e85c2

Many stripe count test

The many stripe count functional test is intended to show that a DNE2 configuration can handle many MDTs in a single filesystem and a single directory can be striped over many MDTs. Due to the virtual AWS environment in which this is being tested, while performance will be measured, neither performance scaling nor load testing are primary goals of this test. It is rather a *functional* scaling test of the ability of the filesystem configuration and directory striping code to handle a large number of MDTs.

1. Create a filesystem with 128 MDTs, 128 OSTs, and at least 128 client mount points (multiple mounts per client)
2. For each stripe count N in 16, 32, 64, 96, 128 create a single striped directory:

```
lfs setdirstripe -c N /mnt/lustre/testN
```

3. Run `mdtest` on all client mount points, and each thread will create/stat/unlink at least 128k files in the striped test directory.
4. No errors will be observed, and balanced striping of files across MDTs will be observed.

Environment Details:

- 8 MDS nodes, each with 16x MDT
- 8 OSS nodes, each with 16x OST
- 8 clients, each with 16 mount points
- All nodes were m3.2xlarge instances
- 4 test runs, each in a single shared 16, 32, 64, 128 striped directory
- mdsrate --create, --stat, --unlink in each directory
- 128k files per MDT for each run
- 8 threads per MDT for each run

Results from AWS showing support up to 128 MDTs:

Stripe count	**16**	**32**	**64**	**128**
Create/sec	21195.2	23149.7	24897.6	22564.1
Stat/sec	30030.6	27263.8	25727.0	28134.7
Unlink/sec	4968.5	5098.8	6198.8	6304.8

Status: **passed locally and on AWS**

Raw results of AWS testing can be found here: https://jira.hpdd.intel.com/browse/LU-6737

Results of testing with 512 simulated stripes can be found in Appendix B.

Migration tool usage documentation

Migration tool is being used to migrate a directory from one MDT to another MDT. The command-line options for the migration tool are:

```
lfs migrate -m <mdt_index> [-v] <migration_dir_path>
-m mdt_index              indicate the MDT index of the target MDT
-v                        show the progress of migration.
migration_dir_path        the name path of the directory being migrated.
```

Note: you can only specify a directory here, i.e. files cannot be migrated independently.

Migrate directory test

1. Setup Lustre with 4 MDTs, 4 OSTs and 1 client.
2. Create 5 directories `/mnt/lustre/migrate{1..5}` and 100 files under each directory on MDT0
3. Create another directory `/mnt/lustre/other_dir` also on MDT0, then create symbol_link/link files, which should be linked to files under `/mnt/lustre/migrate{1..5}`
4. Migrate `/mnt/lustre/migrate{1..5}` from MDT0 to MDT1 by

```
lfs migrate -m 1 /mnt/lustre/migrate1  # migrate migrate1 from current
MDT to MDT1

lfs migrate -m 1 /mnt/lustre/migrate5
```

5. The migration should succeed without errors.
6. Check `/mnt/lustre/migrate{i}`. Multiple link files will still be on MDT0. All other files will be located on MDT1.
7. Verify the mode (permission) of the files and directories is same as before the migration.

Status: **Passed**

Test Results:

https://testing.hpdd.intel.com/sub_tests/9464f282-0e8d-11e5-828a-5254006e85c2

https://testing.hpdd.intel.com/sub_tests/946c315a-0e8d-11e5-828a-5254006e85c2

https://testing.hpdd.intel.com/sub_tests/94757ab2-0e8d-11e5-828a-5254006e85c2

https://testing.hpdd.intel.com/sub_tests/949a6a0c-0e8d-11e5-828a-5254006e85c2

https://testing.hpdd.intel.com/sub_tests/a87eb848-0127-11e5-9d1f-5254006e85c2

https://testing.hpdd.intel.com/sub_tests/a8862164-0127-11e5-9d1f-5254006e85c2

Upgrade with migrate

1. Create a Lustre filesystem with single MDT with Lustre 2.5 and create files and directories as Step **2** and **3** in *Migration Directory Test* above.
2. Upgrade the system to 2.7 + DNE Async Commit patches, and add 3 new MDTs.
3. Complete Steps **4**, **5**, **6**, **7** as described in *Migrate Directory with failed MDT* test below.
4. Run LFSCK to verify filesystem consistency.
5. No errors will be observed.

Status: **Passed as part of conf-sanity 32c (does not include step 4). Passed as documented above in dedicated run on 4th August 2015.**

Conf-Sanity 32c results: https://testing.hpdd.intel.com/test_logs/fe5c679c-2580-11e5-866a-5254006e85c2/show_text

4th August 2015 test results: https://jira.hpdd.intel.com/browse/LU-6955

Migrating directory with failed MDT

1. Setup Lustre with 4 MDTs, 4 OSTs, and 1 client.
2. Create 1 directory (`/mnt/lustre/migrate_dir`) on MDT0 and 100k files within the directory.
3. Start migrating the directory from MDT0 to MDT1. After 30 seconds reboot both MDT0 and MDT1. Note: the reboot must happen during the migration, usually migrating 100k files should take much more than 30 seconds in current 2.6.
    1. `lfs migrate -m 1 -v /mnt/lustre/migrate_dir` # with `-v` you can see the progress of migration.
4. After the MDT0 and MDT1 are restarted and re-mount and recovery finished, client will be able to access the 100k files. Creating files under `/mnt/lustre/migrate_dir` should be denied.
5. Continue the migration with same command:
    1. `lfs migrate -m 1 -v /mnt/lustre/migrate_dir`
6. Check `migrate_dir` and files under `migrate_dir` are located on MDT1.
7. Run LFSCK to verify filesystem consistency.
8. No errors will be present.

Status: **Passed sanity 230c (using slightly different test parameters). Passed as documented above in a dedicated run on 4th Aug 2015.**

Sanity 230C test results: https://testing.hpdd.intel.com/sub_tests/9a79b25c-2580-11e5-866a-5254006e85c2

4th August 2015 test results: https://jira.hpdd.intel.com/browse/LU-6955

Access the directory during migration

1. Setup Lustre with 4 MDTs, 4 OSTs and 2 clients.

2. Create 1 directory and some files under the directory

```
mkdir /mnt/lustre/migrate_dir
for F in {1,2,3,4,5}; do
    echo "$F$F$F$F$F" > /mnt/lustre/migrate_dir/file$F
done
```

3. On one client, migrate the directory among 4 MDTs

```
while true; do
    mdt_idx=$((RANDOM % MDTCOUNT))
    lfs migration -m $mdt_idx /mnt/lustre/migrate_dir || break
done
echo "migrate directory failed"
return 1
```

4. Simultaneously, on another client access these files under the migrating directory

```
while true; do
                ls $migrate_dir2 > /dev/null || {
                        echo "read dir fails"
                        break
                }
                diff -u $DIR2/$tdir/file1 $migrate_dir2/file1 ||{
                        echo "access file1 fails"
                        break
                }
                cat $migrate_dir2/file2 > $migrate_dir2/file3 || {
                        echo "access file2/3 fails"
                        break
                }
                echo"aaaaa">$migrate_dir2/file4 > /dev/null ||{
                        echo "access file4 fails"
                        break
                }

                stat $migrate_dir2/file5 > /dev/null || {
                        echo "stat file5 fails"
                        break
                }
                touch $migrate_dir2/source_file > /dev/null || rc1=$?
                [ $rc1 -ne 0 -o $rc1 -ne 1 ] || {
                        echo "touch file failed with $rc1"
                        break;
                }
                if [ -e $migrate_dir2/source_file ]; then
                        ln $migrate_dir2/source_file $migrate_dir2/link_file \
                                        2&>/dev/null ||rc1=$?
                        if [ -e $migrate_dir2/link_file ]; then
                                rm -rf $migrate_dir2/link_file
                        fi

                        mrename $migrate_dir2/source_file \
                                $migrate_dir2/target_file 2&>/dev/null ||rc1=$?
                        [ $rc1 -ne 0 -o $rc1 -ne 1 ] || {
                                echo "rename failed with $rc1"
                                break
                        }
```

```
                        if [ -e $migrate_dir2/target_file ]; then
                                rm -rf $migrate_dir2/target_file 2&>/dev/null ||
                                        rc1=$?
                        else
                                rm -rf $migrate_dir2/source_file 2&>/dev/null ||
                                        rc1=$?
                        fi
                        [ $rc1 -ne 0 -o $rc1 -ne 1 ] || {
                                echo "unlink failed with $rc1"
                                break
                        }
                fi
        done
```

5. Steps 3 and 4 should keep running at least 5 minutes and will not return error.

Status: **Passed with patch (**http://review.whamcloud.com/14497**) and this test is now included in sanityn.sh as test_80b.**

Link to passing test: https://testing.hpdd.intel.com/sub_tests/770330f6-17f6-11e5-89cc-5254006e85c2

Link to test log: https://testing.hpdd.intel.com/test_logs/2468d136-3766-11e5-9d53-5254006e85c2/show_text

Failover and Recovery Soak Testing

With async update, cross-MDT operations do not need to synchronize updates on each target. Instead, updates are recorded on each target and recovery of the filesystem from failure takes place using these update records. All operations across MDTs are enabled; for example, cross-MDT rename and link succeeds and does not return `-EXDEV`, so a workload like `dbench` that is doing renames should function correctly in a striped directory.

1. Setup Lustre with 4 MDS (each MDS has two MDTs), 4 OSTs, and at least 8 clients.
2. Each client will create a striped directory (stripe_count=4 and default stripe_count=4) so all of their subdirectories will all be striped directories (stripe_count=4), to ensure there are enough cross-MDT operations during the failover test. Under each striped directory,
    1. 3 of 8 clients will keep doing `tar`, `untar` in the striped directory.
    2. 3 of 8 clients will do `dbench` under striped directory.
    3. 2 of 8 clients will run dd (lustre/tests/run_dd.sh)
3. Randomly reboot one of the MDSs at least once every 30 minutes and fail over to the backup MDS if the test configuration allows it. (The optional configuration was not supported on the OpenSFS cluster).
4. The test should keep running at least 24 hours without report application error

Environment details:

- 8 clients (3 running dbench, 3 running tar, and 2 running dd on a striped directory with default stripeEA)
- 4 MDS (each with 1 MDTs)

- 2 OSS (each with 2 OSTs)
- Failover one MDT randomly every 30 mins.
- Passing run on OpenSFS cluster completed on 2nd August 2015 after running for 24 hours and 48 failovers.

Status: **Passed.**

The build that was run to pass the tests is https://build.hpdd.intel.com/job/lustre-reviews/33759/.

The test results and patch list delta from Master can be found in: https://jira.hpdd.intel.com/browse/LU-6773.

The test script used for the tests is: http://review.whamcloud.com/4320

Compatibility Test

Testing of older clients and newer servers, and vice versa, is intended to verify that these combinations fail gracefully when features occur, and do not crash, hang, LBUG, in such situations.

2.5/2.7 Client with 2.7+ patches DNE Async Commit Server

1. Setup Lustre with 4 MDTs, 4 OSTs, and 2 clients. MDT/OST version should be 2.7 + DNE Async Commit. client version should be 2.5.0
2. Run sanity on client with non-striped directories, it should pass.
3. Repeat 1 and 2, but client version will be 2.7.0.

Status: **Passed** Compatibility was demonstrated. There were some failures, however all failures were attributed to non-DNE issues (change in supported features such as removal of SOM code) Details can be found in ticket: https://jira.hpdd.intel.com/browse/LU-6660 and this change records the successful completion: http://review.whamcloud.com/15323

2.7 + patches DNE Async Commit Client with 2.5/2.7 Server

1. Setup Lustre with 4 MDTs, 4 OSTs, and 2 clients. MDT/OST version will be 2.5, client version will be 2.7 + DNE Async Commit.
2. Run sanity on client with non-striped directories. No errors are observed.
3. Repeat 1 and 2 with MDT/OST version 2.7.

Status: **Passed** Compatibility was demonstrated. There were some failures, however all failures were attributed to non-DNE issues (change in supported features such as removal of SOM code) Details can be found in ticket: https://jira.hpdd.intel.com/browse/LU-6661 and this change records the successful completion: http://review.whamcloud.com/15521

Performance Test

The performance testing is intended to measure create and unlink workload scaling as the number of MDS nodes and MDTs is increased. Testing will be done with a single MDS/MDT to provide baseline performance, and then the number of MDTs will be increased to show performance scaling. Testing will be done with both a single MDT per MDS as well as multiple MDTs per MDS to explore likely deployment configurations in production environments.

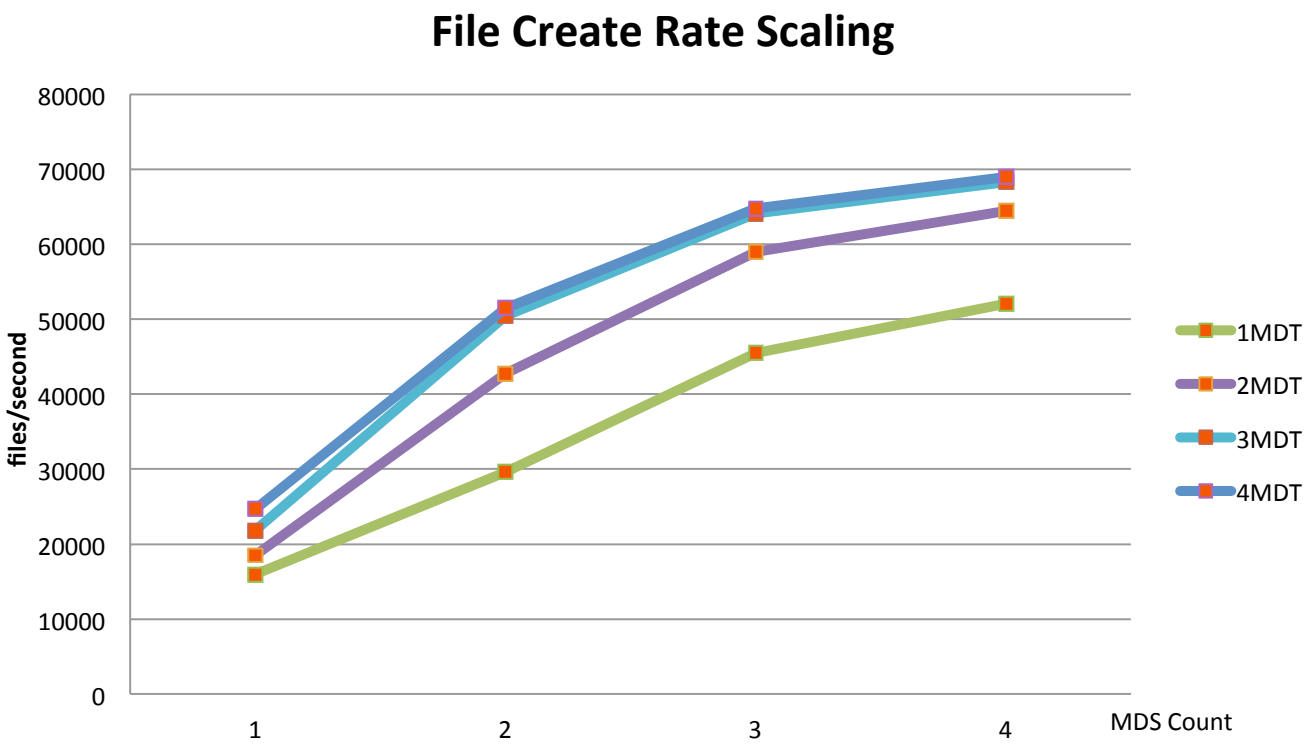
1. Configure Lustre with 1, 2, 3, then 4 MDS nodes, each with 1, 2, 3, 4 MDTs per MDS, with the number of clients available on the OpenSFS test cluster.
2. Configure clients with multiple mount points to allow more concurrent metadata RPCs
3. For each test create a single directory striped over all MDTs
4. Create 20,000 files per client within the striped directory, measure aggregate create rate.
5. Unlink all files created by each client, measure aggregate unlink rate

Status: **Passed**

Due to hardware limitations, the configuration available on the OpenSFS test cluster was:

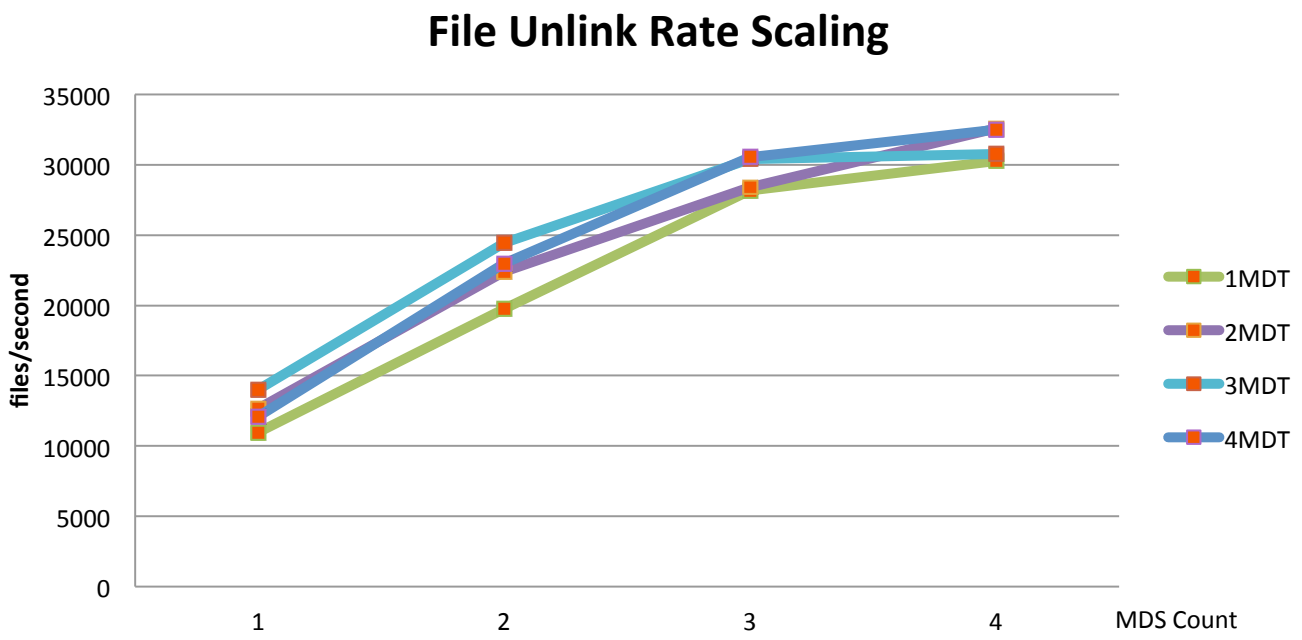
- 20 clients, one mountpoint per client
  - 24 threads per client
  - fail_loc=0x804 to allow multiple modify RPC per mount point
- 4 MDS (4 MDTs on each MDS)
  - MDT 20GB ldiskfs OSD on 1TB SATA HDD
  - 10GB external journal on 90GB SATA SSD
- 1 OSS (8 OSTs)
  - OST 50GB ldiskfs OSD on 1TB SATA HDD, internal journal
- QDR Infiniband network between all nodes

In initial performance testing on Lustre 2.7.0, a lock bottleneck was found in the kernel quota layer that prevented file create performance scaling with multiple MDTs on the same node with ldiskfs. This has largely been avoided in osd-ldiskfs, but needs further kernel-side changes in order to make the kernel quota more scalable. As a result, for these tests, quota has been disabled in ldiskfs.



The mdsrate File Create Rate Scaling graph shows that striping the directory over additional MDTs on new MDS nodes (constant number of MDTs per MDS) improves the file creation rate approximately linearly with each new MDS - between 81-93% per additional MDS. The scaling shown when adding MDTs on an MDS (constant MDS Count) shows a reasonable improvement for a second MDT on the same MDS, approximately 30-45%, but only 10-20% for the third MDT, and virtually no performance improvement for the fourth MDT.

The lack of continued scaling when adding MDTs on an MDS is likely a result of two related factors. Internal Lustre and kernel locking limit the scalability of a single MDS node, so adding more MDTs to the same MDS increases contention on the same locks. Secondly, the hardware resources of the MDS node are limited (CPU, RAM, network) so increasing the MDT count does not allow further scaling without adding more hardware. Since the Lustre MDT code is largely multi-threaded already, and no performance degradation was observed from 3 MDTs/MDS to 4 MDTs/MDS the scaling limit is likely the result of limited hardware resources.



Similar to the File Create results, the mdsrate File Unlink Rate Scaling performance shows good scaling improvements by striping the directory over additional MDTs on separate MDS nodes - between 81-95% improvement for each added MDS. The scaling shown by adding MDTs on an MDS is relatively modest for unlinks, only about 10% improvement for the second MDT, and a variable amount for the third and fourth MDTs, in some cases showing negative scaling for three and four MDTs.

This suggests that for optimal performance scaling with DNE striped directories that there is little benefit for having more than two MDTs per MDS. Since there is still some noticeable performance improvement from having two MDTs per MDS, and this more fully utilizes the MDS hardware, it is reasonable to run in this configuration. For upgrading existing active-passive two MDS configurations, it would be most cost effective to add three new MDTs to the existing MDS nodes and could achieve a 105-165% performance improvement for a relatively minimal increase in hardware. Raw results from this work is recorded in Appendix C: Performance results raw data and on https://jira.hpdd.intel.com/browse/LU-6786

Appendices

Appendix A: Optional Test: Stress testing with `racer.sh`

The primary test for migration + file operations is covered under *Access the directory during Migration*. As an optional test racer + migrate was attempted with the below steps.

Racer is a stress testing framework designed to test races and unusual combinations of operations that would not otherwise be caught by regular regression test scripts. It creates files, directories, hard and symbolic links, then performs random operations thereon. It is easily extensible to add in new types of file and namespace operations. For testing DNE 2 functionality a racer script to create a randomly-named remote striped directories with random MDT index and random stripe counts is added. A racer script to migrate a randomly-named file or directory to a random MDT index is also added.

1. Create a filesystem with 4 MDTs, 4 OSTs and 2 clients.
2. Run `racer` with MDSCOUNT=4, striped directory.
3. Racer will attempt to run for 10 minutes without LBUG.
4. Clients and MDSes should be able to unmount without problems.
5. Run LFSCK to verify filesystem consistency.

Status: Failed. We ran into issues with this test including an Assertion that we believe is related to the `LOHA_EXISTS` setting in `osp_object_getattr()`. We plan to continue to look into and resolve these issues as part of Lustre 2.8 stabilization.

In order to verify that directory migration is working under normal usage circumstances, additional tests were performed with migrate combined with other common operations such as cat, append, ls, ln, rm, stat, touch, rename, as shown in the *Access the directory during Migration* section.

Appendix B: Simulated Wide Striping

Results of testing with 512 simulated stripes. This is using a special testing hook (fail_loc=0x1703) that allows multiple stripes to be allocated from the same MDT in order to allow functional testing with limited HW. This can be seen in the output below, which re-uses the same MDT indices repeatedly, and allocates multiple objects (0x104 through 0x183 = 128) from the same FID sequence on each MDT.

While this functionality does not exercise all of the same code paths as a fully-configured testbed, it does allow exercising some of these wide striping code paths on a continual basis with a simple test configuration.

```
== sanity test 300k: test large striped directory == 01:30:04 (1433320204)
fail_loc=0x1703
fail_loc=0
/mnt/lustre/d300k.sanity/striped_dir
lmv_stripe_count: 512
lmv_stripe_offset: 0
mdtidx           FID[seq:oid:ver]
     0          [0x200000400:0x104:0x0]
     1          [0x240000403:0x104:0x0]
     2          [0x280000403:0x104:0x0]
     3          [0x2c0000403:0x104:0x0]
     0          [0x200000400:0x105:0x0]
     1          [0x240000403:0x105:0x0]
     2          [0x280000403:0x105:0x0]
     3          [0x2c0000403:0x105:0x0]
     0          [0x200000400:0x106:0x0]
     1          [0x240000403:0x106:0x0]
     2          [0x280000403:0x106:0x0]
     3          [0x2c0000403:0x106:0x0]
     0          [0x200000400:0x107:0x0]
     1          [0x240000403:0x107:0x0]
     2          [0x280000403:0x107:0x0]
     3          [0x2c0000403:0x107:0x0]
     0          [0x200000400:0x108:0x0]
     1          [0x240000403:0x108:0x0]
     2          [0x280000403:0x108:0x0]
     3          [0x2c0000403:0x108:0x0]
     0          [0x200000400:0x109:0x0]
     1          [0x240000403:0x109:0x0]
     2          [0x280000403:0x109:0x0]
     3          [0x2c0000403:0x109:0x0]
     0          [0x200000400:0x10a:0x0]
     1          [0x240000403:0x10a:0x0]
     2          [0x280000403:0x10a:0x0]
     3          [0x2c0000403:0x10a:0x0]
     0          [0x200000400:0x10b:0x0]
     1          [0x240000403:0x10b:0x0]
     2          [0x280000403:0x10b:0x0]
     3          [0x2c0000403:0x10b:0x0]
     0          [0x200000400:0x10c:0x0]
     1          [0x240000403:0x10c:0x0]
     2          [0x280000403:0x10c:0x0]
     3          [0x2c0000403:0x10c:0x0]
     0          [0x200000400:0x10d:0x0]
     1          [0x240000403:0x10d:0x0]
     2          [0x280000403:0x10d:0x0]
     3          [0x2c0000403:0x10d:0x0]
     0          [0x200000400:0x10e:0x0]
     1          [0x240000403:0x10e:0x0]
     2          [0x280000403:0x10e:0x0]
     3          [0x2c0000403:0x10e:0x0]
     0          [0x200000400:0x10f:0x0]
     1          [0x240000403:0x10f:0x0]
```

```
2           [0x280000403:0x10f:0x0]
3           [0x2c0000403:0x10f:0x0]
0           [0x200000400:0x110:0x0]
1           [0x240000403:0x110:0x0]
2           [0x280000403:0x110:0x0]
3           [0x2c0000403:0x110:0x0]
0           [0x200000400:0x111:0x0]
1           [0x240000403:0x111:0x0]
2           [0x280000403:0x111:0x0]
3           [0x2c0000403:0x111:0x0]
0           [0x200000400:0x112:0x0]
1           [0x240000403:0x112:0x0]
2           [0x280000403:0x112:0x0]
3           [0x2c0000403:0x112:0x0]
0           [0x200000400:0x113:0x0]
1           [0x240000403:0x113:0x0]
2           [0x280000403:0x113:0x0]
3           [0x2c0000403:0x113:0x0]
0           [0x200000400:0x114:0x0]
1           [0x240000403:0x114:0x0]
2           [0x280000403:0x114:0x0]
3           [0x2c0000403:0x114:0x0]
0           [0x200000400:0x115:0x0]
1           [0x240000403:0x115:0x0]
2           [0x280000403:0x115:0x0]
3           [0x2c0000403:0x115:0x0]
0           [0x200000400:0x116:0x0]
1           [0x240000403:0x116:0x0]
2           [0x280000403:0x116:0x0]
3           [0x2c0000403:0x116:0x0]
0           [0x200000400:0x117:0x0]
1           [0x240000403:0x117:0x0]
2           [0x280000403:0x117:0x0]
3           [0x2c0000403:0x117:0x0]
0           [0x200000400:0x118:0x0]
1           [0x240000403:0x118:0x0]
2           [0x280000403:0x118:0x0]
3           [0x2c0000403:0x118:0x0]
0           [0x200000400:0x119:0x0]
1           [0x240000403:0x119:0x0]
2           [0x280000403:0x119:0x0]
3           [0x2c0000403:0x119:0x0]
0           [0x200000400:0x11a:0x0]
1           [0x240000403:0x11a:0x0]
2           [0x280000403:0x11a:0x0]
3           [0x2c0000403:0x11a:0x0]
0           [0x200000400:0x11b:0x0]
1           [0x240000403:0x11b:0x0]
2           [0x280000403:0x11b:0x0]
3           [0x2c0000403:0x11b:0x0]
0           [0x200000400:0x11c:0x0]
1           [0x240000403:0x11c:0x0]
2           [0x280000403:0x11c:0x0]
3           [0x2c0000403:0x11c:0x0]
0           [0x200000400:0x11d:0x0]
1           [0x240000403:0x11d:0x0]
2           [0x280000403:0x11d:0x0]
3           [0x2c0000403:0x11d:0x0]
0           [0x200000400:0x11e:0x0]
1           [0x240000403:0x11e:0x0]
2           [0x280000403:0x11e:0x0]
3           [0x2c0000403:0x11e:0x0]
0           [0x200000400:0x11f:0x0]
1           [0x240000403:0x11f:0x0]
2           [0x280000403:0x11f:0x0]
3           [0x2c0000403:0x11f:0x0]
0           [0x200000400:0x120:0x0]
1           [0x240000403:0x120:0x0]
```

```
2            [0x280000403:0x120:0x0]
3            [0x2c0000403:0x120:0x0]
0            [0x200000400:0x121:0x0]
1            [0x240000403:0x121:0x0]
2            [0x280000403:0x121:0x0]
3            [0x2c0000403:0x121:0x0]
0            [0x200000400:0x122:0x0]
1            [0x240000403:0x122:0x0]
2            [0x280000403:0x122:0x0]
3            [0x2c0000403:0x122:0x0]
0            [0x200000400:0x123:0x0]
1            [0x240000403:0x123:0x0]
2            [0x280000403:0x123:0x0]
3            [0x2c0000403:0x123:0x0]
0            [0x200000400:0x124:0x0]
1            [0x240000403:0x124:0x0]
2            [0x280000403:0x124:0x0]
3            [0x2c0000403:0x124:0x0]
0            [0x200000400:0x125:0x0]
1            [0x240000403:0x125:0x0]
2            [0x280000403:0x125:0x0]
3            [0x2c0000403:0x125:0x0]
0            [0x200000400:0x126:0x0]
1            [0x240000403:0x126:0x0]
2            [0x280000403:0x126:0x0]
3            [0x2c0000403:0x126:0x0]
0            [0x200000400:0x127:0x0]
1            [0x240000403:0x127:0x0]
2            [0x280000403:0x127:0x0]
3            [0x2c0000403:0x127:0x0]
0            [0x200000400:0x128:0x0]
1            [0x240000403:0x128:0x0]
2            [0x280000403:0x128:0x0]
3            [0x2c0000403:0x128:0x0]
0            [0x200000400:0x129:0x0]
1            [0x240000403:0x129:0x0]
2            [0x280000403:0x129:0x0]
3            [0x2c0000403:0x129:0x0]
0            [0x200000400:0x12a:0x0]
1            [0x240000403:0x12a:0x0]
2            [0x280000403:0x12a:0x0]
3            [0x2c0000403:0x12a:0x0]
0            [0x200000400:0x12b:0x0]
1            [0x240000403:0x12b:0x0]
2            [0x280000403:0x12b:0x0]
3            [0x2c0000403:0x12b:0x0]
0            [0x200000400:0x12c:0x0]
1            [0x240000403:0x12c:0x0]
2            [0x280000403:0x12c:0x0]
3            [0x2c0000403:0x12c:0x0]
0            [0x200000400:0x12d:0x0]
1            [0x240000403:0x12d:0x0]
2            [0x280000403:0x12d:0x0]
3            [0x2c0000403:0x12d:0x0]
0            [0x200000400:0x12e:0x0]
1            [0x240000403:0x12e:0x0]
2            [0x280000403:0x12e:0x0]
3            [0x2c0000403:0x12e:0x0]
0            [0x200000400:0x12f:0x0]
1            [0x240000403:0x12f:0x0]
2            [0x280000403:0x12f:0x0]
3            [0x2c0000403:0x12f:0x0]
0            [0x200000400:0x130:0x0]
1            [0x240000403:0x130:0x0]
2            [0x280000403:0x130:0x0]
3            [0x2c0000403:0x130:0x0]
0            [0x200000400:0x131:0x0]
1            [0x240000403:0x131:0x0]
```

```
2            [0x280000403:0x131:0x0]
3            [0x2c0000403:0x131:0x0]
0            [0x200000400:0x132:0x0]
1            [0x240000403:0x132:0x0]
2            [0x280000403:0x132:0x0]
3            [0x2c0000403:0x132:0x0]
0            [0x200000400:0x133:0x0]
1            [0x240000403:0x133:0x0]
2            [0x280000403:0x133:0x0]
3            [0x2c0000403:0x133:0x0]
0            [0x200000400:0x134:0x0]
1            [0x240000403:0x134:0x0]
2            [0x280000403:0x134:0x0]
3            [0x2c0000403:0x134:0x0]
0            [0x200000400:0x135:0x0]
1            [0x240000403:0x135:0x0]
2            [0x280000403:0x135:0x0]
3            [0x2c0000403:0x135:0x0]
0            [0x200000400:0x136:0x0]
1            [0x240000403:0x136:0x0]
2            [0x280000403:0x136:0x0]
3            [0x2c0000403:0x136:0x0]
0            [0x200000400:0x137:0x0]
1            [0x240000403:0x137:0x0]
2            [0x280000403:0x137:0x0]
3            [0x2c0000403:0x137:0x0]
0            [0x200000400:0x138:0x0]
1            [0x240000403:0x138:0x0]
2            [0x280000403:0x138:0x0]
3            [0x2c0000403:0x138:0x0]
0            [0x200000400:0x139:0x0]
1            [0x240000403:0x139:0x0]
2            [0x280000403:0x139:0x0]
3            [0x2c0000403:0x139:0x0]
0            [0x200000400:0x13a:0x0]
1            [0x240000403:0x13a:0x0]
2            [0x280000403:0x13a:0x0]
3            [0x2c0000403:0x13a:0x0]
0            [0x200000400:0x13b:0x0]
1            [0x240000403:0x13b:0x0]
2            [0x280000403:0x13b:0x0]
3            [0x2c0000403:0x13b:0x0]
0            [0x200000400:0x13c:0x0]
1            [0x240000403:0x13c:0x0]
2            [0x280000403:0x13c:0x0]
3            [0x2c0000403:0x13c:0x0]
0            [0x200000400:0x13d:0x0]
1            [0x240000403:0x13d:0x0]
2            [0x280000403:0x13d:0x0]
3            [0x2c0000403:0x13d:0x0]
0            [0x200000400:0x13e:0x0]
1            [0x240000403:0x13e:0x0]
2            [0x280000403:0x13e:0x0]
3            [0x2c0000403:0x13e:0x0]
0            [0x200000400:0x13f:0x0]
1            [0x240000403:0x13f:0x0]
2            [0x280000403:0x13f:0x0]
3            [0x2c0000403:0x13f:0x0]
0            [0x200000400:0x140:0x0]
1            [0x240000403:0x140:0x0]
2            [0x280000403:0x140:0x0]
3            [0x2c0000403:0x140:0x0]
0            [0x200000400:0x141:0x0]
1            [0x240000403:0x141:0x0]
2            [0x280000403:0x141:0x0]
3            [0x2c0000403:0x141:0x0]
0            [0x200000400:0x142:0x0]
1            [0x240000403:0x142:0x0]
```

```
2           [0x280000403:0x142:0x0]
3           [0x2c0000403:0x142:0x0]
0           [0x200000400:0x143:0x0]
1           [0x240000403:0x143:0x0]
2           [0x280000403:0x143:0x0]
3           [0x2c0000403:0x143:0x0]
0           [0x200000400:0x144:0x0]
1           [0x240000403:0x144:0x0]
2           [0x280000403:0x144:0x0]
3           [0x2c0000403:0x144:0x0]
0           [0x200000400:0x145:0x0]
1           [0x240000403:0x145:0x0]
2           [0x280000403:0x145:0x0]
3           [0x2c0000403:0x145:0x0]
0           [0x200000400:0x146:0x0]
1           [0x240000403:0x146:0x0]
2           [0x280000403:0x146:0x0]
3           [0x2c0000403:0x146:0x0]
0           [0x200000400:0x147:0x0]
1           [0x240000403:0x147:0x0]
2           [0x280000403:0x147:0x0]
3           [0x2c0000403:0x147:0x0]
0           [0x200000400:0x148:0x0]
1           [0x240000403:0x148:0x0]
2           [0x280000403:0x148:0x0]
3           [0x2c0000403:0x148:0x0]
0           [0x200000400:0x149:0x0]
1           [0x240000403:0x149:0x0]
2           [0x280000403:0x149:0x0]
3           [0x2c0000403:0x149:0x0]
0           [0x200000400:0x14a:0x0]
1           [0x240000403:0x14a:0x0]
2           [0x280000403:0x14a:0x0]
3           [0x2c0000403:0x14a:0x0]
0           [0x200000400:0x14b:0x0]
1           [0x240000403:0x14b:0x0]
2           [0x280000403:0x14b:0x0]
3           [0x2c0000403:0x14b:0x0]
0           [0x200000400:0x14c:0x0]
1           [0x240000403:0x14c:0x0]
2           [0x280000403:0x14c:0x0]
3           [0x2c0000403:0x14c:0x0]
0           [0x200000400:0x14d:0x0]
1           [0x240000403:0x14d:0x0]
2           [0x280000403:0x14d:0x0]
3           [0x2c0000403:0x14d:0x0]
0           [0x200000400:0x14e:0x0]
1           [0x240000403:0x14e:0x0]
2           [0x280000403:0x14e:0x0]
3           [0x2c0000403:0x14e:0x0]
0           [0x200000400:0x14f:0x0]
1           [0x240000403:0x14f:0x0]
2           [0x280000403:0x14f:0x0]
3           [0x2c0000403:0x14f:0x0]
0           [0x200000400:0x150:0x0]
1           [0x240000403:0x150:0x0]
2           [0x280000403:0x150:0x0]
3           [0x2c0000403:0x150:0x0]
0           [0x200000400:0x151:0x0]
1           [0x240000403:0x151:0x0]
2           [0x280000403:0x151:0x0]
3           [0x2c0000403:0x151:0x0]
0           [0x200000400:0x152:0x0]
1           [0x240000403:0x152:0x0]
2           [0x280000403:0x152:0x0]
3           [0x2c0000403:0x152:0x0]
0           [0x200000400:0x153:0x0]
1           [0x240000403:0x153:0x0]
```

```
2            [0x280000403:0x153:0x0]
3            [0x2c0000403:0x153:0x0]
0            [0x200000400:0x154:0x0]
1            [0x240000403:0x154:0x0]
2            [0x280000403:0x154:0x0]
3            [0x2c0000403:0x154:0x0]
0            [0x200000400:0x155:0x0]
1            [0x240000403:0x155:0x0]
2            [0x280000403:0x155:0x0]
3            [0x2c0000403:0x155:0x0]
0            [0x200000400:0x156:0x0]
1            [0x240000403:0x156:0x0]
2            [0x280000403:0x156:0x0]
3            [0x2c0000403:0x156:0x0]
0            [0x200000400:0x157:0x0]
1            [0x240000403:0x157:0x0]
2            [0x280000403:0x157:0x0]
3            [0x2c0000403:0x157:0x0]
0            [0x200000400:0x158:0x0]
1            [0x240000403:0x158:0x0]
2            [0x280000403:0x158:0x0]
3            [0x2c0000403:0x158:0x0]
0            [0x200000400:0x159:0x0]
1            [0x240000403:0x159:0x0]
2            [0x280000403:0x159:0x0]
3            [0x2c0000403:0x159:0x0]
0            [0x200000400:0x15a:0x0]
1            [0x240000403:0x15a:0x0]
2            [0x280000403:0x15a:0x0]
3            [0x2c0000403:0x15a:0x0]
0            [0x200000400:0x15b:0x0]
1            [0x240000403:0x15b:0x0]
2            [0x280000403:0x15b:0x0]
3            [0x2c0000403:0x15b:0x0]
0            [0x200000400:0x15c:0x0]
1            [0x240000403:0x15c:0x0]
2            [0x280000403:0x15c:0x0]
3            [0x2c0000403:0x15c:0x0]
0            [0x200000400:0x15d:0x0]
1            [0x240000403:0x15d:0x0]
2            [0x280000403:0x15d:0x0]
3            [0x2c0000403:0x15d:0x0]
0            [0x200000400:0x15e:0x0]
1            [0x240000403:0x15e:0x0]
2            [0x280000403:0x15e:0x0]
3            [0x2c0000403:0x15e:0x0]
0            [0x200000400:0x15f:0x0]
1            [0x240000403:0x15f:0x0]
2            [0x280000403:0x15f:0x0]
3            [0x2c0000403:0x15f:0x0]
0            [0x200000400:0x160:0x0]
1            [0x240000403:0x160:0x0]
2            [0x280000403:0x160:0x0]
3            [0x2c0000403:0x160:0x0]
0            [0x200000400:0x161:0x0]
1            [0x240000403:0x161:0x0]
2            [0x280000403:0x161:0x0]
3            [0x2c0000403:0x161:0x0]
0            [0x200000400:0x162:0x0]
1            [0x240000403:0x162:0x0]
2            [0x280000403:0x162:0x0]
3            [0x2c0000403:0x162:0x0]
0            [0x200000400:0x163:0x0]
1            [0x240000403:0x163:0x0]
2            [0x280000403:0x163:0x0]
3            [0x2c0000403:0x163:0x0]
0            [0x200000400:0x164:0x0]
1            [0x240000403:0x164:0x0]
```

```
2            [0x280000403:0x164:0x0]
3            [0x2c0000403:0x164:0x0]
0            [0x200000400:0x165:0x0]
1            [0x240000403:0x165:0x0]
2            [0x280000403:0x165:0x0]
3            [0x2c0000403:0x165:0x0]
0            [0x200000400:0x166:0x0]
1            [0x240000403:0x166:0x0]
2            [0x280000403:0x166:0x0]
3            [0x2c0000403:0x166:0x0]
0            [0x200000400:0x167:0x0]
1            [0x240000403:0x167:0x0]
2            [0x280000403:0x167:0x0]
3            [0x2c0000403:0x167:0x0]
0            [0x200000400:0x168:0x0]
1            [0x240000403:0x168:0x0]
2            [0x280000403:0x168:0x0]
3            [0x2c0000403:0x168:0x0]
0            [0x200000400:0x169:0x0]
1            [0x240000403:0x169:0x0]
2            [0x280000403:0x169:0x0]
3            [0x2c0000403:0x169:0x0]
0            [0x200000400:0x16a:0x0]
1            [0x240000403:0x16a:0x0]
2            [0x280000403:0x16a:0x0]
3            [0x2c0000403:0x16a:0x0]
0            [0x200000400:0x16b:0x0]
1            [0x240000403:0x16b:0x0]
2            [0x280000403:0x16b:0x0]
3            [0x2c0000403:0x16b:0x0]
0            [0x200000400:0x16c:0x0]
1            [0x240000403:0x16c:0x0]
2            [0x280000403:0x16c:0x0]
3            [0x2c0000403:0x16c:0x0]
0            [0x200000400:0x16d:0x0]
1            [0x240000403:0x16d:0x0]
2            [0x280000403:0x16d:0x0]
3            [0x2c0000403:0x16d:0x0]
0            [0x200000400:0x16e:0x0]
1            [0x240000403:0x16e:0x0]
2            [0x280000403:0x16e:0x0]
3            [0x2c0000403:0x16e:0x0]
0            [0x200000400:0x16f:0x0]
1            [0x240000403:0x16f:0x0]
2            [0x280000403:0x16f:0x0]
3            [0x2c0000403:0x16f:0x0]
0            [0x200000400:0x170:0x0]
1            [0x240000403:0x170:0x0]
2            [0x280000403:0x170:0x0]
3            [0x2c0000403:0x170:0x0]
0            [0x200000400:0x171:0x0]
1            [0x240000403:0x171:0x0]
2            [0x280000403:0x171:0x0]
3            [0x2c0000403:0x171:0x0]
0            [0x200000400:0x172:0x0]
1            [0x240000403:0x172:0x0]
2            [0x280000403:0x172:0x0]
3            [0x2c0000403:0x172:0x0]
0            [0x200000400:0x173:0x0]
1            [0x240000403:0x173:0x0]
2            [0x280000403:0x173:0x0]
3            [0x2c0000403:0x173:0x0]
0            [0x200000400:0x174:0x0]
1            [0x240000403:0x174:0x0]
2            [0x280000403:0x174:0x0]
3            [0x2c0000403:0x174:0x0]
0            [0x200000400:0x175:0x0]
1            [0x240000403:0x175:0x0]
```

```
     2           [0x280000403:0x175:0x0]
     3           [0x2c0000403:0x175:0x0]
     0           [0x200000400:0x176:0x0]
     1           [0x240000403:0x176:0x0]
     2           [0x280000403:0x176:0x0]
     3           [0x2c0000403:0x176:0x0]
     0           [0x200000400:0x177:0x0]
     1           [0x240000403:0x177:0x0]
     2           [0x280000403:0x177:0x0]
     3           [0x2c0000403:0x177:0x0]
     0           [0x200000400:0x178:0x0]
     1           [0x240000403:0x178:0x0]
     2           [0x280000403:0x178:0x0]
     3           [0x2c0000403:0x178:0x0]
     0           [0x200000400:0x179:0x0]
     1           [0x240000403:0x179:0x0]
     2           [0x280000403:0x179:0x0]
     3           [0x2c0000403:0x179:0x0]
     0           [0x200000400:0x17a:0x0]
     1           [0x240000403:0x17a:0x0]
     2           [0x280000403:0x17a:0x0]
     3           [0x2c0000403:0x17a:0x0]
     0           [0x200000400:0x17b:0x0]
     1           [0x240000403:0x17b:0x0]
     2           [0x280000403:0x17b:0x0]
     3           [0x2c0000403:0x17b:0x0]
     0           [0x200000400:0x17c:0x0]
     1           [0x240000403:0x17c:0x0]
     2           [0x280000403:0x17c:0x0]
     3           [0x2c0000403:0x17c:0x0]
     0           [0x200000400:0x17d:0x0]
     1           [0x240000403:0x17d:0x0]
     2           [0x280000403:0x17d:0x0]
     3           [0x2c0000403:0x17d:0x0]
     0           [0x200000400:0x17e:0x0]
     1           [0x240000403:0x17e:0x0]
     2           [0x280000403:0x17e:0x0]
     3           [0x2c0000403:0x17e:0x0]
     0           [0x200000400:0x17f:0x0]
     1           [0x240000403:0x17f:0x0]
     2           [0x280000403:0x17f:0x0]
     3           [0x2c0000403:0x17f:0x0]
     0           [0x200000400:0x180:0x0]
     1           [0x240000403:0x180:0x0]
     2           [0x280000403:0x180:0x0]
     3           [0x2c0000403:0x180:0x0]
     0           [0x200000400:0x181:0x0]
     1           [0x240000403:0x181:0x0]
     2           [0x280000403:0x181:0x0]
     3           [0x2c0000403:0x181:0x0]
     0           [0x200000400:0x182:0x0]
     1           [0x240000403:0x182:0x0]
     2           [0x280000403:0x182:0x0]
     3           [0x2c0000403:0x182:0x0]
     0           [0x200000400:0x183:0x0]
     1           [0x240000403:0x183:0x0]
     2           [0x280000403:0x183:0x0]
     3           [0x2c0000403:0x183:0x0]
Resetting fail_loc on all nodes...done.
PASS 300k (7s)
== sanity test complete, duration 9 sec == 01:30:11 (1433320211)
```

Appendix C: Performance results raw data

20 clients (with fail_loc=0x804, and each client run 24 threads, totally 480 threads) with 4 MDSes (each MDS has up to 4 MDTs) and 1 OSS (8 OSTs)

```
usr/lib64/openmpi/bin/mpirun -np 480 -machinefile /home/di.wang/machine_file
/usr/lib64/lustre/tests/mdsrate --mknod --dir /mnt/lustre/test_n --filefmt
'f%%d' --nfiles 960000
```

Mknod

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS
1	15972	18478	21825	24668
2	29616	42737	50412	51461
3	45520	58967	64138	64805
4	52034	64459	68294	68985

Unlink

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS
1	10978	12669	14021	12091
2	19772	22410	24451	22976
3	28184	28407	30382	30555
4	30256	32575	30750	32475

An additional 5 test runs produced the following data:

Mknod Mean Aggregate Rates

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS
1	15796.2	18366.9	21725.5	24106.4
2	28920.7	38091.9	49249.8	46674.8
3	48948.9	57828.1	54075.2	65179.2
4	55374.2	57107.5	68028.9	68683.7

Unlink Mean Aggregate Rates

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS

1	11113.9	12750.6	12600.1	12650.5
2	22038.0	22124.9	23061.7	24074.1
3	28579.4	25158.9	29239.0	30165.9
4	28063.6	28590.3	32760.1	33432.3

Mknod Standard Deviation

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS
1	189.614	248.172	82.983	471.721
2	314.631	7943.809	2026.755	11382.423
3	2234.408	2892.442	18932.353	1035.150
4	3592.449	20703.777	372.576	731.603

Unlink Standard Deviation

MDS count	1 MDT/MDS	2 MDT/MDS	3 MDT/MDS	4 MDT/MDS
1	111.063	145.657	662.592	736.033
2	2171.947	1457.276	1471.920	805.600
3	1550.384	5957.475	677.554	495.668
4	6557.413	7042.628	524.715	248.097