# Testing a Lustre filesystem



Lustre code available from Whamcloud git repository contains tools to test a Lustre installation. Since creation of Lustre in 2001, these tools have matured and multiplied. To date, 22 different test suits are regularly run and many more are available.

## Test suite overview

This document assumes that you have a Linux kernel compiled with Lustre patches. Typical routes to getting a working Lustre kernel include:

- By downloading a pre-build kernel from a provider.
- By applying the Lustre patches and building your own kernel.
  Details on both of these routes is provided on the wiki page: Putting together a Lustre filesystem.

### Pre-requisites

The instructions on this page assume that you have the Lustre test suite installed. You can get this from source at http://git.whamcloud.com/ or as RPM from server builds at build.whamcloud.com.

### Configuring `llmount.sh`

`llmount.sh` is configured from environment variables stored in the `$LUSTRE/tests/cfg/` directory by default. By default, the test configuration for a test system is in the `local.sh` file. If you want to specify configuration values other than the defaults (e.g. to specify block devices instead of loopback files, change the number of OSTs being tested, etc), the best way to do this is to set the `NAME={lustre_fsname}` variable and create a new configuration script `$LUSTRE/tests/cfg/${lustre_fsname}` that contains the site-specific variables, and sources `$LUSTRE/tests/cfg/local.sh` for the default values. This allows keeping site-specific values separate from files that are in Git or RPM packages, but ensures that any new or release-specific changes to `local.sh` are used (if not already specified). For example, a test configuration for a developer running on a single node might look like:

```
$ export NAME=testfs                                    # used by test-framework.sh to find the
configuration file
$ cat lustre/tests/cfg/$NAME.sh
FSNAME=testfs
MDSDEVBASE=/dev/vg_testfs/lvmdt
OSTDEVBASE=/dev/vg_testfs/lvost
OSTCOUNT=${OSTCOUNT:-5}
MODOPTS_LIBCFS="libcfs_panic_on_lbug=0"
FAIL_ON_ERROR=${FAIL_ON_ERROR:-true}
export SHARED_DIRECTORY="/tmp"                        # /tmp is shared for all services on the test node
. $LUSTRE/tests/cfg/local.sh                        # source all of the other configuration defaults
unset OSTSIZE                                                 # use the size of the lvost devices, not a
fixed size
unset MDSSIZE                                                 # use the size of the lvmdt devices, not a
fixed size
```

### `llmount.sh`

One of the simplest test suites consists of `llmount.sh` (to format and mount a test filesystem) and `llmountcleanup.sh` (to unmount the filesystem). `llmount.sh` uses a collection of bash scripts to create a Lustre file system complete with MDS, MDT, OSS, OST and Client using loop devices on a single machine. `llmountcleanup.sh` tears down the work `llmount.sh` performed and should return your system to normal.

Once `llmount.sh` has completed successfully you should see the following:

```
[root@client-10 ~/lustre-git]# cd lustre/tests
[root@client-10 ~/lustre-git/lustre/tests]# ./llmount.sh
Stopping clients: client-10.lab.whamcloud.com /mnt/lustre (opts:)
Stopping clients: client-10.lab.whamcloud.com /mnt/lustre2 (opts:)
Loading modules from /build/lustre-release/lustre/tests/..
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet options: 'networks=tcp0 accept=all'
Formatting mgs, mds, osts
Checking servers environments
Checking clients client-10.lab.whamcloud.com environments
Setup mgs, mdt, osts
Starting mds: -o loop  /tmp/lustre-mdt /mnt/mds
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=256
Started lustre-MDT0000
Starting ost1: -o loop  /tmp/lustre-ost1 /mnt/ost1
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=256
Started lustre-OST0000
Starting ost2: -o loop  /tmp/lustre-ost2 /mnt/ost2
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=256
Started lustre-OST0001
Starting client: client-10.lab.whamcloud.com: -o user_xattr,acl,flock client-10.lab.whamcloud.com@tcp:/lustre
/mnt/lustre
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=256
Using TIMEOUT=20
[root@client-10 ~/lustre-git/lustre/tests]#
```

## Troubleshooting `llmount.sh`

**`llmount.sh` falls over complaining that it cannot connect to the server**

This error has multiple possible causes. One common problem is that the system `hostname` maps to `localhost` (`127.0.0.1`) instead of a real IP address. Another possible problem is when testing on a single node using the Infiniband (IB) network. Even though `llmount.sh` does not connect to any external machines the IB network must be working correctly. It is possible to switch to TCP for the purposes of running `llmount.sh` - select network using `NETTYPE=tcp` in the config file and check that LNet is configured to use tcp in `/etc/modprobe.d/lustre.conf`. More details on LNet are available in the manual.

**`llmount.sh` complains that a value is undefined**

Before you run `llmount.sh` it is necessary to set the debug size environment variable in the configuration, for example `DEBUG_SIZE=256`. Setting `DEBUG_SIZE` to this value ensures enough space is allocated for logs for all the CPUs in the system. If `DEBUG_SIZE` is too small, the param setting will complain during `llmount.sh`

1. You will now have a lustre filesystem available to you in user-space at `/mnt/lustre/` or `/mnt/$FSNAME/` if it was specified differently in your configuration.
2. You can test this by switching striping to all nodes and writing a big file:

```
[root@client-10 ~/lustre-git/lustre/tests]# lfs setstripe -c -1 /mnt/lustre
[root@client-10 ~/lustre-git/lustre/tests]# lfs getstripe /mnt/lustre/
/mnt/lustre/
stripe_count:   -1 stripe_size:    0 stripe_offset:  -1
[root@client-10 ~/lustre-git/lustre/tests]# dd if=/dev/zero of=/mnt/lustre/file.out bs=1MB count=400
400+0 records in
400+0 records out
400000000 bytes (400 MB) copied, 2.33261 seconds, 171 MB/s
```

3. Clean-up the after the tests:

```
./llmountcleanup.sh
```

## auster

Auster is a test-framework that runs a large suite of functional tests for Lustre. There is good coverage of all Lustre functionality contained within Auster. Help is available on-line:

```
$ /usr/lib64/lustre/tests/auster -h
Usage auster [options]  suite [suite options] [suite [suite options]]
Run Lustre regression tests suites.
      -c CONFIG Test environment config file
      -d LOGDIR Top level directory for logs
      -D FULLLOGDIR Full directory for logs
      -f STR    Config name (cfg/<name>.sh)
      -g GROUP  Test group file (Overrides tests listed on command line)
      -S TESTSUITE First test suite to run allows for restarts
      -H        Honor the EXCEPT and ALWAYS_EXCEPT list when --only is used
      -i N      Repeat tests N times (default 1). A new directory
                will be created under LOGDIR for each iteration.
      -k        Don't stop when subtests fail
      -R        Remount lustre between tests
      -r        Reformat (during initial configuration if needed)
      -s        SLOW=yes
      -v        Verbose mode
      -l        Send logs to the Maloo database after run
                  (can be done later by running maloo_upload.sh)
      -h        This help.

Suite options
These are suite specific options that can be specified after each suite on
the command line.
   suite-name  [options]
      --only LIST         Run only specific list of subtests
      --except LIST       Skip list of subtests
      --start-at SUBTEST  Start testing from subtest
      --stop-at SUBTEST   Stop testing at subtest
      --time-limit LIMIT  Don't allow this suite to run longer
                          than LIMT seconds. [UNIMPLEMENTED]


Example usage:
Run all of sanity and all of replay-single except for 70b with SLOW=y using
the default "local" configuration.

  auster -s sanity replay-single --except 70b

Run all tests in the regression group 5 times using large config.

  auster -f large -g test-groups/regression -i 5
```

## Run tests using auster script

- Single node

```
# cd /usr/lib64/lustre/tests
# ./auster -rv runtests
```

Note: This is a very simple setup, not all tests can be run in this configuration

- Multiple nodes

```
# cd /usr/lib64/lustre/tests
edit cfg/testfs.sh
Minimum required variables: mds_HOST, ost_HOST, PDSH, MDSDEV1, OSTCOUNT, OSTDEV#, MDS_MOUNT_OPTS, OST_MOUNT_OPTS


See Lustre Test Tools Environment Variable for more information
Make sure partitions on the disks are setup
If using real devices, make sure to set MDS_MOUNT_OPTS, OST_MOUNT_OPTS = ""
If there is more than one clients set RCLIENTS=<list of remote clients>


# ./auster -rvf testfs runtests (or any other test suite)
```

Test logs will be in /tmp/test_logs/YYYY-MM-DD.

Subsequence runs do not need to reformat (-r option) the filesystem.

## Run Individual Tests with auster

By default, auster will run all tests in a test suite unless a test is explicitly set not to run or a user specifies a subset of tests to run.

Given a test suite abc.sh with tests 1a, 1b, 1c, 1d, 1e, 2a, 2b, 2c, 2d, 2e, 3, 4, 5a, 5b, 6, 7a 7b, 7c, 11, 12, 13, 14, 21, 22, 23a, 23b, 24, 25

```
./auster abc
```

will run all tests in suite abc.

If you want to run a single or small number of tests, you can use the test suite flag “--only”. No matter what order you put the tests in, they will be run in numerical order. For example

```
./auster abc --only "1b 2c 25 6"
./auster abc --only "1b,2c,25,6"
./auster abc --only 1b,2c,25,6
```

will run abc.sh tests 1b, 2c, 6, and 25; note that test 6 will be run before test 25.

If the "--only" flag is used more than once for the same test suite, only the tests specified in the last "--only" flag will be run.

```
./auster abc --only "1b,2c” –only “25,6"
```

will only run abc.sh tests 6 and 25. Tests 1b and 2c are not run.

Many Lustre test suites have subtests that test the same feature or functionality. These tests are numbered with an integer followed by a letter. If a test suite has tests with the same number with different letters appended, then you can run all tests with the same number, all letters, by specifying only the number in the "--only" list of tests.

```
./auster abc --only 2
```

will run abc.sh tests 2a, 2b, 2c, 2d, and 2e.

By default, using the "--only" flag overrides the ALWAYS_EXCEPT list in the test suite script and will execute a test in the "--only" list even if the same test is on the ALWAYS_EXCEPT list. For example, assume abc.sh test suite has ALWAYS_EXCEPT=”2c 3 4” defined, running

```
./auster abc --only "2 3"
```

will run abc.sh tests 2a, 2b, 2c, 2d, 2e, and 3 even though test 2c and test 3 are on the ALWAYS_EXCEPT list.

Auster provides a way to override this default behavior of ignoring the ALWAYS_EXCEPT list when using --only by using the '-H' flag.

```
./auster -H abc --only "2 3"
```

will run abc.sh tests 2a, 2b, 2d and 2e only. The ALWAYS_EXCEPT list is honored and tests 2c and 3 are skipped.

If you specify a test that does not exist in the "--only" list of tests to run, no warning or error messages are printed. For example, running

```
./auster abc --only v4d
```

will not complain that test 4d does not exist.

## Run a Range of Tests with auster

### Range of tests using --only

If you want to run a range of tests, you can use the test suite flag "--only".

```
./auster abc --only "11-14"
./auster abc --only 11-14
```

will run abc.sh tests 11, 12, 13, and 14.

The test number at the beginning and/or end of the range does not have to exist in the test suite; all tests in a range that exist are run. For example,

```
./auster abc --only 10-15
```

will run abc.sh tests 11, 12, 13, and 14.

Spaces between the hyphen are not allowed and will give an error and will not run any tests.

```
./auster abc --only “10 – 15”
```

will run no tests.

More than one range can be specified. The ranges will be combined and tests will be run in numerical order.

```
./auster abc --only "11-14 3-4"
./auster abc --only "11-14,3-4"
./auster abc --only 11-14,3-4
```

will run abc.sh tests in the order of 3, 4, 11, 12, 13, and 14.

Ranges can overlap and each test will only be run once even if it is specified in more than one range.

```
./auster abc --only "11-13,12-14"
```

will run abc.sh tests 11, 12, 13, and 14.

If the –only flag is used more than once for the same test suite, only the tests specified in the range of the last –only flag will be run.

```
./auster abc --only 3-4 --only 12-14
```

will run abc.sh tests 12, 13, and 14. Tests 3 and 4 will not be run.

If a test suite has tests with the same number with letters appended, then all tests with the same number all letters will be run if the test number falls into a range.

```
./auster abc --only 6-10
```

will run abc.sh tests 6, 7a 7b, and 7c.

Using a range with the --only flag overrides the ALWAYS_EXCEPT list in the test suite script and will execute a test in the –only list even if the same test is on the ALWAYS_EXCEPT list. For example, assume abc.sh test suite has ALWAYS_EXCEPT=”2c 3 4” defined, running

```
./auster abc --only 3-5
```

will run abc.sh tests 3, 4, 5a, and 5b even though tests 3 and 4 are on the ALWAYS_EXCEPT list.

The --only flag does not allow a range to start with a subtest, a number followed by a letter, and does not allow a range to end with a subtest. All of the following will get an error “seq: invalid floating point argument: 2d” or “seq: invalid floating point argument: 5d” and no tests will be run. The following are NOT supported

```
./auster abc --only “2d-5a”
./auster abc --only “2d-4”
./auster abc --only “3-5d”
```

### Range of tests using --start-at and --stop-at

The per test suite options --start-at and --stop-at currently do not change what tests are run and should not be used.

```
./auster abc --start-at 3 --stop-at 6
```

will run all tests in abc.sh ignoring the --start-at and --stop-at flags.

Using the tests individually do not control starting or stopping of executing tests and should not be used individually. The following do notchange what tests are run and should NOT be used

```
./auster abc --start-at 3
./auster abc --stop-at 6
```

## Exclude Individual Tests with auster

If you do not want to run a test or a small number of tests, auster provides the --except flag to exclude tests from being run.

```
 ./auster -k -v abc --except 3
```

will run all tests in abc.sh except for test 3.

If a test has subtests, the --except flag can be used to skip all subtests just by specifying the test number.

```
./auster -k -v abc --except 1
```

will run all tests in test suite abc.sh except for test 1 and will not run all subtests of 1. Thus, 1a, 1b, 1c, 1d, and 1e will not be run, but all other tests will executed. Also, tests listed on the ALWAYS_EXCEPT list are also skipped.

Skipping individual subtests is also supported.

```
./auster -k -v abc --except 1b
```

will run all tests in test suite abc.sh except for test 1b. Thus, 1a, 1c, 1d, and 1e will be run along with all other tests  greater than 1 in abc.sh.

```
 ./auster -k -v abc --except 1,2,3,4,5,6,7
```

will skip tests 1 through 7 and all sub tests.

## Exclude Ranges of Tests with auster

Specifying a range of tests to exclude does not work and no tests will be executed. The following does NOT work

```
./auster -k -v abc --except 1-7
./auster -k -v abc --except “1-7”
```