whamcloud

# Feature Test Plan
# for
# Wide Striping

Revision History

| Date | Revision | Author | Description of Document Change |
|---|---|---|---|
| 12/08/2011 | A | Yu Jian | First draft. |
| 02/06/2012 | B | Yu Jian | Second draft.<br>Add upgrade/downgrade testing. |
| | | | |
| | | | |

# Table of Contents

# 1 Introduction

## 1.1 Objectives

This test plan is intended to conduct the testing of Lustre wide striping feature. The objectives are:

1) To identify the items to be tested.
2) To define the test cases and approaches.
3) To define the test pass/fail criteria.
4) To define the test deliverables.

## 1.2 Background

Currently a file can be stripped across OSTs up to a limit of 160 stripes due to the ldiskfs xattr size limit of 4096 bytes. The wide striping feature expands that stripe count limit to 2000 stripes or more by supporting large xattrs in ldiskfs. This feature effectively increases the maximum file size under ldiskfs to 2000 * 16TB = 32PB.

## 1.3 References

- Lustre file striping across large number of OSTs - presentation by Oleg Drokin http://www.olcf.ornl.gov/wp-content/events/lug2011/4-14-2011/1100-1130_Oleg_Drokin_LUG2011-Widestriping.pdf

# 2 Test items

The wide striping feature was implemented by making changes to the following components:

1) ldiskfs: http://review.whamcloud.com/1708
   This patch implements the large xattr support in ldiskfs. If the size of an xattr value is larger than the blocksize, then the xattr value would not be saved in the external xattr block, instead it would be saved in an external xattr inode. So, the patch also helps support a larger number of xattrs.

2) lov: http://review.whamcloud.com/1111
   This patch expands the Lustre stripe count limit from 160 to 2000 and it is possible to go even to larger stripe count.

3) mds: http://review.whamcloud.com/1808 and http://review.whamcloud.com/1930
   The first patch implements dynamic RPC buffer and the second one fixes open resent issue with last_xid.

4) utils: http://review.whamcloud.com/1194
   This patch allows Lustre clients to interoperate with servers that do not support large xattrs, and also servers with support with at least 2000 stripes, though it is intended not to have any hard upper limit.

5) tests: http://review.whamcloud.com/1880
   This patch adds and updates some test cases in auster test suite to verify the large xattr feature. To enable this feature, the "-O large_xattr" option needs to be set on the filesystem either with --mkfsoptions at format time or via tune2fs.

6) e2fsprogs: https://bugzilla.lustre.org/attachment.cgi?id=30751, https://bugzilla.lustre.org/attachment.cgi?id=20372 and http://review.whamcloud.com/1886
   These patches implement the large xattr support in e2fsprogs.

The large xattr support in ldiskfs component could be tested independently. The “tests” component depends on “lov” and “ldiskfs” changes. All of the components would be tested with e2fsprogs version 1.41.90.wc4.

# 3 Approach

Currently the large xattr feature is disabled by default at mkfs.lustre time. To test the wide striping feature, the "-O large_xattr" option needs to be set on the filesystem either with --mkfsoptions at format time or via tune2fs.

## 3.1 Regression testing

A full Lustre auster test suite should be performed by the autotest system to ensure that the applied wide striping changes on Lustre master branch do not adversely affect previously tested functionality.

The typical configuration of the autotest system to run a full auster test suite is:

```
OSSCOUNT=1
OSTCOUNT=7
MDSCOUNT=1
CLIENTCOUNT=2

ENABLE_QUOTA=yes
SLOW=yes
NETTYPE=tcp

TEST_GROUP=full
```

The “full” test group contains the following tests:

```
runtests, sanity, sanity-benchmark, sanityn, lfsck, liblustre, conf-sanity, racer, replay-single, recovery-small, replay-ost-single, replay-dual, replay-vbr, insanity, sanity-quota, sanity-sec, lustre-rsync-test, metadata-updates, ost-pools, mmp, performance-sanity, parallel-scale, large-scale, lnet-selftest, obdfilter-survey, sgpdd-survey
```

It usually takes about 24 hours to finish a full auster test run on Toro cluster by the autotest system.

## 3.2 Functional testing

### 3.2.1 Large xattrs

The large xattr feature is supported in the ldiskfs component, which could be tested independently.

There are two test suites to be used:

1) xfstests
   The xfstests #020 (operates on extended attributes) and #062 (exercises the getfattr/setfattr tools) are xattrs specific tests.

   By applying the following two patches from the linux-ext4 list, xfstests #020 and #062 would pass on ext4 filesystem:

   ```
   [PATCH V2] xfstests: 020: make this xattr test generic
   [PATCH V2] xfstests: Sort recursive getfattr output in 062
   ```

   And then by making some small changes, the tests would work on the ldiskfs filesystem and support testing over 4kB xattr value. Both #020 and #062 tests should pass:

   ```
   # mkfs.lustre --reformat --mdt --mgs --fsname=lustre1 --mkfsoptions="-O large_xattr" /dev/sdb1
   # mkfs.lustre --reformat --mdt --mgs --fsname=lustre2 --mkfsoptions="-O large_xattr" /dev/sdb2
   # mkdir -p /mnt/mds1 && mount -t ldiskfs -o user_xattr,acl /dev/sdb1 /mnt/mds1
   # mkdir -p /mnt/mds2 && mount -t ldiskfs -o user_xattr,acl /dev/sdb2 /mnt/mds2

   # ATTRVAL_SIZE=65536 TEST_DIR=/mnt/mds1 TEST_DEV=/dev/sdb1 SCRATCH_MNT=/mnt/mds2
   SCRATCH_DEV=/dev/sdb2 ./check 020 062
   020 1s ... 1s
   062 1s ... 2s
   Ran: 020 062
   Passed all 2 tests
   ```

2) auster
Some basic functional tests for large xattrs to verify their correct operations should be added/updated in the Lustre auster test suite. These tests are:

| |
|---|
| sanity test 102ha "grow xattr from inside inode to external inode" |
| conf-sanity test 61 "large xattr" |
| replay-single test 58b "test replay of setxattr op" |
| replay-single test 58c "resend/reconstruct setxattr op" |
| lustre-rsync-test test 1 "Simple Replication" |

Please refer to http://review.whamcloud.com/1880 for test details.

### 3.2.2 Large stripe count

Since the wide striping feature increases the maximum stripe count from 160 to 2000. The large stripe count should be tested.

The following configuration could be used for the Lustre auster test suite to verify the large stripe count support:

```
OSSCOUNT=2
OSTCOUNT=2000 (with 1000 OSTs per OSS)
MDSCOUNT=1
CLIENTCOUNT=2

MDSOPT="--mkfsoptions='-O large_xattr -J size=4096' --mountfsoptions='errors=remount-ro,user_xattr,acl'"

ENABLE_QUOTA=yes
SLOW=yes
NETTYPE=o2ib
```

And at least the following tests (stripe count is set to -1) in the auster test suite should be performed:

| |
|---|
| sanity-benchmark test bonnie "bonnie++" |
| sanity-benchmark test iozone "iozone" |
| sanity-benchmark test fsx "fsx" |
| parallel-scale test iorssf "IOR test with single-shared-file mode" |
| parallel-scale test iorfpp "IOR test with file-per-process mode" |

## 3.3 Performance testing

To test the performance, a patched mdtest (adds option [-x xattr_size]) could be used to create and get xattrs: http://jira.whamcloud.com/secure/attachment/10665/mdtest_xattr.patch

```
# mkfs.lustre --reformat --mdt --mgs --mkfsoptions="-O large_xattr" /dev/sdb1
# mkdir -p /mnt/mds1 && mount -t ldiskfs -o user_xattr,acl /dev/sdb1 /mnt/mds1

# for i in 100 1000 5000; do mkdir -p /mnt/mds1/mdtest$i && ./mdtest -u -d /mnt/mds1/mdtest$i/ -F -i 10 -n 10000 -x $i; done
```

The existing large xattr patch for ldiskfs in http://review.whamcloud.com/1708 saves the xattr value in an external inode if the size of the value is larger than the blocksize. This would introduce metadata performance degradation for operating large xattrs due to the need to create an extra inode for any xattr over 4kB.

There is a plan to implement "mid-sized" xattrs (up to 64kB) by referencing them directly from the xattr block pointer (LU-908). This was requested by the upstream ext4 maintainer before accepting the large xattr patches into mainline Linux. The 64kB "mid-sized" xattrs are not included in the current patch set. After it is implemented, the performance would be improved.

## 3.4 Interoperability testing

The wide striping feature will debut in Lustre 2.2 release. To ensure the previous Lustre versions which do not support large stripe count still work with the Lustre 2.2 servers, the interoperability testing needs to be done.

The following two combinations need to be tested:

1) Lustre 1.8.7-wc1 clients and 2.2 servers
2) Lustre 2.1.0 clients and 2.2 servers

And the following tests need to be performed:

1) A full auster test suite should be performed by the autotest system to ensure there is no regression issue. Please refer to section 3.1 for the typical configuration.

2) Intermix different Lustre versions as follows:

```
OSS1: version 2.2 (with 1000 OSTs)
OSS2: version 2.2 (with 1000 OSTs)
MGS/MDS: version 2.2 (with “large_xattr” enabled)
CLIENT1: version 2.2
CLIENT2: version 2.1.0
CLIENT3: version 1.8.7-wc1
```

   a) Mount the large stripe count (2000 OSTs in total) filesystem on Lustre 1.8.7-wc1, 2.1.0 and 2.2 clients.

   b) Since the “utils” changes are not in Lustre 1.8.7-wc1 and 2.1.0, running “lfs setstripe” on CLIENT2 and CLIENT3 would not allow creating a file or setting the stripe count on a directory with more than 160 stripes:

```
# lfs setstripe -c 161 /mnt/lustre/testfile
error: bad stripe count 161: Invalid argument (22)
error: setstripe: create stripe file '/mnt/lustre/testfile' failed

# mkdir -p /mnt/lustre/testdir
# lfs getstripe /mnt/lustre/testdir
/mnt/lustre/testdir
stripe_count:   0 stripe_size:    0 stripe_offset:  -1
# lfs setstripe -c 161 /mnt/lustre/testdir
error: bad stripe count 161: Invalid argument (22)
error: setstripe: create stripe file '/mnt/lustre/testdir' failed
```

   c) Run “lfs setstripe” on CLIENT1 to create a file with 2000 stripes, then running “lfs getstripe” on CLIENT2 and CLIENT3 should correctly get the striping information and running IOR should successfully write and read the wide-striped file.

   d) Run IOR on CLIENT1 to create and write a single shared file with 2000 stripes, then running “lfs getstripe” on CLIENT2 and CLIENT3 should correctly get the striping information and running IOR should successfully read the wide-striped file.

   e) Create a directory on CLIENT1 with 2000 stripes, then running IOR from CLIENT2 and CLIENT3 should successfully create, write and read a single shared file with 2000 stripes under the directory.

   f) Remove the wide-striped files from CLIENT2 or CLIENT3. The files should be successfully unlinked and the filesystem space should be freed.

## 3.5 Upgrade/downgrade testing

Since the previous Lustre versions do not support wide striping, the following upgrade/downgrade paths need to be tested:

1) upgrade from Lustre 1.8.7-wc1 to 2.2 (with “large_xattr” enabled), then downgrade to 1.8.7-wc1
2) upgrade from Lustre 2.1.0 to 2.2 (with “large_xattr” enabled), then downgrade to 2.1.0

After MDS is upgraded, "tune2fs -O large_xattr /dev/mdtdev" should be run on the MDT to enable wide striping feature. Then, once this feature is enabled and in use on the MDT, it will not be possible to directly downgrade the MDT filesystem to the earlier version of Lustre that does not support wide striping. The only way to disable it would be to delete (or use "lfs_migrate" with a stripe_count = 160) all of the files with large xattrs and then unmount the MDT and then run "tune2fs -O ^large_xattr /dev/mdtdev" to turn off this filesystem feature.

# 4 Item pass/fail criteria

- The test items must pass all tests.
- The metadata performance degradation is allowed.

# 5 Test deliverables

- This test plan will be used to conduct the testing of wide striping feature.
- New regression tests will be added into the Lustre auster test suite.
- Defects found during the testing will be reported to JIRA.
- Test results will be uploaded to Maloo.