Lustre Autotest and Maloo

Andreas Dilger <adilger@Whamcloud.com>
Whamcloud Development Infrastructure

► JIRA Issue Tracking  https://jira.whamcloud.com/
► Git  https://git.whamcloud.com/
► Gerrit Change Review  https://review.whamcloud.com/
► Jenkins Build Server  https://build.whamcloud.com/
► Maloo Test Results Repo  https://testing.whamcloud.com/

► Q & A
Multiple Separate Lustre Development Systems

► **JIRA for issue tracking (bugs, features, improvements) for code and docs**
  • Gerrit posts initial patch submission, landing automatically to JIRA

► **Gerrit tracks patches and reviews for all Lustre projects**
  • Whamcloud developers have an internal OpenID provider
  • Community developers use Google OAuth2 or other OpenID provider
  • Use Settings->Link Another Identity to add new identity to account

► **Jenkins for automated build**
  • servers: `el8.6 x86_64`
  • clients: `el7.9, el8.6, e19.0, sles15sp4, ubuntu2004; x86_64, aarch64, ppcle64`

► **Multiple Autotest test sessions launched for successful patch build**
  • All *enforced* sessions must pass
  • The *optional sessions do not need to pass*

► **Maloo tracks test results from Autotest**
  • Posts into Gerrit when all sessions are started
  • Posts into Gerrit when session is complete
  • Marks patch **Verified: -1** if *any enforced session* fails
Autotest Test Sessions

Maloo

Patch Set 14:
The following sessions will be run for build 89569 patch 14

- 20 or more test sessions per patch!
  - Multiple sessions **enforced** for master patches
  - Multiple sessions **optional** for arches, distros

- Enforced test sessions must pass
  - Custom Test-Parameters: runs are **enforced**

- Optional sessions are **optional**
  - Run to improve coverage before all tests pass
  - Working toward passing and enforcement
  - Do not resubmit optional sessions

- Janitor only -1 for failures in modified tests
- Please use test resources wisely
  - 170 tests, 13500 subtests, 152h **per patch**

Maloo Test Queue: https://testing.whamcloud.com/test_queue?jobs=lustre-reviews&builds=89569&commit=Apply+Filter
Maloo Results: https://testing.whamcloud.com/test_sessions?jobs=lustre-reviews&builds=89569&start_date=2022-09-15#redirect
Control Autotest Sessions with Test-Parameters

Test-Parameters: testlist=sanityn env=ONLY=16,HONOR_EXCEPT=y
Test-Parameters: serverversion=2.14 testlist=sanityn env=ONLY=16,HONOR_EXCEPT=y

Fixes: 7905359296 ("LU-15748 osc: fallbackate interop for 2.14 clients")
Signed-off-by: Andreas Dilger <adilger@whamcloud.com>
Change-Id: I327183025a8de6fd814a7c2929365497153ebbe5

► Use the Test-Parameters: tag in a commit message:
  • Test-Parameters: testlist=sanity,racer # run additional test scripts, one session per line
  • ... env=ONLY=32,ONLY_REPEAT=50 # set env vars (run specific subtest many times)
  • ... ostfilesystemtype=zfs # use specific backing fstype
  • ... clientversion=2.12.9 # old client new server interop
  • ... serverversion=2.14 # new client old server interop
  • ... mdscount=4 mdtcount=8 ostcount=8 # change server counts for DNE testing

► Use trivial keyword only for limited testing test scripts, man pages, build, etc.
  • ... trivial # runs only sanity, Inet-selftest, now also aarch64
  • ... testlist=recovery-small,replay-single # supplement with sessions, configs as needed

https://wiki.whamcloud.com/display/PUB/Changing+Test+Parameters+with+Gerrit+Commit+Messages
Adding Follow-Up Test Sessions Easily (New)

- [Follow-Up] creates a new empty patch
- Add Test-Parameters as needed
- Good to add test sessions to existing patches
- Useful when patch already finished testing
- Merge Test-Parameters into patch if refresh
- Patch marked "Work-In-Progress", cannot land
<table>
<thead>
<tr>
<th>Gerrit Patch Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>jenkins</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Lustre G... Janitor</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Lustre G... Janitor</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
<tr>
<td>Maloo</td>
</tr>
</tbody>
</table>

- Custom tests are numbered in order starting at custom-101
- Lustre Gerrit Janitor is a separate test system that runs under loads that provoke race conditions
- Janitor test sessions may finish quickly and supply early feedback
- Janitor only adds Verified -1 for failures on tests added/modified in a patch, but failures should still be reviewed for correctness
- Optional test sessions are not required to pass for a test to land
Monitoring Test Results in Maloo

Patch Set 9:
The following sessions will be run for build 89713 patch 9

Enforced:
- review-idiskfs-dne on el8.5-x86_64/idiskfs servers and clients
- review-idiskfs-dne-arm on el8.5-x86_64/idiskfs servers and el8.5-aarch64/idiskfs clients
- custom on el8.5-x86_64/idiskfs servers and clients

Maloo Test Queue: https://testing.whamcloud.com/test_queue?jobs=lustre-reviews&builds=89713&commit=Apply+Filter
Maloo Results: https://testing.whamcloud.com/test_sessions?jobs=lustre-reviews&builds=89713&start_date=2022-09-25#redirect

<table>
<thead>
<tr>
<th>Job</th>
<th>Build/Path</th>
<th>Seq/Idx</th>
<th>Test Group</th>
<th>Distro server</th>
<th>Review</th>
<th>Suite/Test</th>
<th>Started</th>
</tr>
</thead>
</table>
| lustre-reviews | 89713:9  | 1:28    | review-idiskfs-dne | server: el8.5-x86_64/idiskfs clients: el8.5-x86_64/idiskfs | 73232604d | sanity test_31j | 2022-09-25 21:24
|           |           |         |                  |                |         |                | (~1 hour)        |
| lustre-reviews | 89713:9  | 1:29    | review-idiskfs-dne-arm | server: el8.5-x86_64/idiskfs clients: el8.5-aarch64/idiskfs | 73232604d | sanity test_24u | 2022-09-25 21:24
|           |           |         |                  |                |         |                | (~1 hour)        |
| lustre-reviews | 89713:9  | 1:101   | custom           | server: el8.5-x86_64/idiskfs clients: el8.5-x86_64/idiskfs | 73232604d | conf-sanity test_20 | 2022-09-25 21:24
|           |           |         |                  |                |         |                | (~1 hour)        |

- Permanently keeps results for all patches
- Logs are kept as long as possible (months), longer for test failures if space available
Reviewing Test Failures in Maloo

Error: 'buffered io, not write rpc: grants mismatch: 12656640, expected 4218880'

Failure Rate: 1.05% of most recent 95 runs, 5 skipped (all branches)

OST 1, OST 2, OST 3, OST 4, OST 5, OST 6, OST 7, OST 8 (trevis-100vm9)
  • sanity.test_64f.debug.trevis-100vm9.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.debug_log.trevis-100vm9.1664078772.log [ Download | Show | Preview 50 ]

MDS 1, MDS 3 (trevis-100vm10)
  • sanity.test_64f.debug.trevis-100vm10.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.debug_log.trevis-100vm10.1664078772.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.dmesg.trevis-100vm10.1664078772.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.dmesg.trevis-100vm10.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.journal.trevis-100vm10.log [ Download | Show | Preview 50 ]
  • sanity.test_64f.stack_trace.trevis-100vm10.log [ Download | Show | Preview 50 ]

Bugs
No bugs

► Review test failures
► Associate known failures to existing bugs
► Raise new bug for new test failures
► Help track common failures/rates/trends
► Quick search for subtest failures by clicking on failed subtest name
Search Test Results in Maloo

Start search from Results menu
- Select Test or Subtest search

May need to increase date range to find occasional failures for subtest

Check for other similar failures with associated LU ticket to speed up failure analysis

Some failures are caused by prior errors (e.g. 'mount failed')

Repeated failures for same patch or ticket indicate patch regression

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Date</th>
<th>Duration</th>
<th>Code</th>
<th>Error</th>
<th>Links</th>
<th>Test log</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-25</td>
<td>3604</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:47131, jira:LU-8151</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-20</td>
<td>3602</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:47121, jira:LU-15777</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-19</td>
<td>10</td>
<td>1</td>
<td>direct io with grant allocation: grant</td>
<td>gerrit:48679, jira:LU-18165</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-13</td>
<td>14</td>
<td>1</td>
<td>direct io with grant allocation: grant</td>
<td>gerrit:43707, jira:LU-14139</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-13</td>
<td>3602</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:48278, jira:LU-16087</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-07</td>
<td>3601</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:48445, jira:EX-5398</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-05</td>
<td>3601</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:45690, jira:LU-15058</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-09-02</td>
<td>3</td>
<td>1</td>
<td>mount failed</td>
<td>gerrit:46334, jira:EX-4567</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-08-31</td>
<td>3</td>
<td>1</td>
<td>mount failed</td>
<td>gerrit:46334, jira:EX-4567</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-08-29</td>
<td>3</td>
<td>1</td>
<td>mount failed</td>
<td>gerrit:46334, jira:EX-4567</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-08-29</td>
<td>3603</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:48278, jira:LU-16087</td>
<td>Preview 50</td>
</tr>
<tr>
<td>test_64f</td>
<td>FAIL</td>
<td>2022-08-29</td>
<td>3603</td>
<td>1</td>
<td>buffered io, not write rpc: grants mi</td>
<td>gerrit:48278, jira:LU-16087</td>
<td>Preview 50</td>
</tr>
</tbody>
</table>
Associate Failures with Multiple Patches

While reviewing your own patch, also associate the same LU ticket for other similar test failures:

- Helps other developers
- Improves failure analysis
Limit Maloo Search Incrementally

- Very useful when looking for test regressions
- Limit or increase search dates to find when failure started
  - Broad searches with large date ranges/many results will timeout, limit searches to specific subtest and status
- Limit branch, architecture, filesystem to isolate effects
- "git log --oneline --after YYYY-MM-DD --before YYYY-MM-DD" to find related patch landings

Search

Test sets | Sub tests | Bug links
Keep Patches Moving Forward

► Developers are responsible for their own patches
  • Update patches based on inspections
  • Keep inspections and testing moving forward

► Provide inspection feedback for others in a timely manner

► After inspection, help move patch to next state
  • Two or more inspectors on the patch?
  • Tests have passed or triaged and retested as needed?

► Good inspections avoid more effort finding defects later
  • Chance to find and fix latent bugs, cleanup code
Autotest Failures Affect Everyone

- Need to identify all test failures on a patch before resubmitting
  - Patch may otherwise be adding new intermittent bugs

- **Associate** or **Raise** bugs in Maloo for all test failures

- Search for and triage similar failures via Maloo, JIRA
  - Can be done more efficiently in batches

- **Data needed to prioritize bugs causing the most failures**
  - This speeds up fixing the most frequent failures
  - This in turn speeds up every other patch testing and landing
Using Git and Gerrit
Managing Branches

- Keep patches for independent changes on separate branches
- A git "branch" is just a pointer to some series of commits
- Each commit or rebase moves pointer to new tip of the branch
- All of the old commits/branches stay in repository, unreferenced
  - `git checkout branch` # check out specific branch
  - `git branch {new_branch}` # create new, keep current branch
  - `git checkout -b {new_branch}` # create new branch and move there
  - `git status` # show modified/staged/untracked
Finding Branches

► Show local branches

```
# git branch -lv
b_test5  5fefbd1078 LU-15548 tests: skip conf-sanity/131 for older servers
b_def_mask  5efdbbeab24 LU-9859 libcfs: add "default" keyword for debug mask
b_comma_list  76cffe0f2c LU-12597 tests: remove need for comma_list
master  1b09fb2296 LU-16062 ldlm: improve bl_timeout for prolong
```

► Show all remote branches

```
# git branch -alv
[as above, plus]
remotes/origin/HEAD-> origin/master
remotes/prime/b2_13  a298f8c137 New release 2.13.0
remotes/prime/b2_14  e1635c6afb New release 2.14.0
```
Searching in Git

- `git diff [-w]`  
  # show current change (no whitespace changes)
- `git diff --cached`  
  # show "git add" changes that are staged for commit
- `git log [branch|path]`  
  # show commits (branch/dir/file)
- `git log -S {string}`  
  # search commits in patches that add/remove specific string
- `git log --grep {regexp}`  
  # search commit messages with regular expression
- `git grep {regexp}`  
  # show all files in repository with regular expression
- `git log --patch`  
  # print patch/diff with each log message
- `git show [-w] [commit]`  
  # show specific commit (no whitespace)
- `git describe [commit]`  
  # name [of commit] relative to tag (e.g. v2_14_57-26-g54d4cca6)
Finding Change History of Code

► Shows history of commits that modified some line number
  
  # git blame lustre/obdclass/class_obd.c
  
  52257a67e4d (Liu Xuezha0 2012-07-10 21:13:26 +0800 96)
  829dd8b39a8 (Nathan Rutman 2009-05-28 00:00:48 +0000 97)
  5707f919e41 (James Simmons 2013-09-27 10:24:42 -0400 99)
  5707f919e41 (James Simmons 2013-09-27 10:24:42 -0400 100)
  54d4cca6cb0 (Arshad Hussain 2021-03-21 06:32:07 +0530 101)

► Show history of commits before a specific commit
  
  # git blame lustre/obdclass/class_obd.c {commit}~1

► Show patches landed between specific dates
  
  # git log --after 2022-09-10 --before 2022-09-12
Gerrit Searching

Search shortcut/bookmark in Firefox

Edit bookmark

Name

Patch Review

URL

https://review.whamcloud.com/#/q/%s

Tags

Separate tags with commas

Use tags to organize and search for bookmarks from the address bar

Keyword

review

Use a single keyword to open bookmarks directly from the address bar

Cancel  Save

Searching for patches is very flexible

• A few basic keywords are useful to know
• Many options available (state, content, age, reviews)
• Default to search for word in commit message

• open change #12345
  review 12345
• find commit messages for specific text
  review LU-12250
• find merged patches owned by specific user
  review owner:bzzz is:merged
• find open patches modifying specific file
  review file:lustre_idl.h is:open

https://review.whamcloud.com/Documentation/user-search.html
Commit Message Should Describe Patch Well

- Patch summary line should be useful by itself
  - Appears in many patch summaries ("git log --oneline", "git reflog", etc.)
- What should be in the commit message?
  - What problem/feature does this patch address?
  - Usage/test that triggered problem, with error messages/stack
  - Performance improvements resulting from patch
  - How it is being fixed, clear description of tricky code
- Fixes: label finds related patches when searching
- Why is the commit message content important?
  - Initial inspectors understand the patch more quickly
  - Gatekeeper understands importance of the patch
  - Anyone dealing with the patch later has some guidance

https://wiki.whamcloud.com/display/PUB/Commit+Comments
Committing Changes

► Install Git commit hooks for patch verification before push to Gerrit
  
  # ln -sf ../../contrib/git-hooks/commit-msg .git/hooks
  # ln -sf ../../contrib/git-hooks/prepare-commit-msg .git/hooks

► Read the comments provided by prepare-commit-msg hook!

  # git add [path ...]  # stage specified file(s) or dirs for commit
  # git add [-u]        # stage all changes for commit
  # git commit -av      # stage and commit all changes, show diff
  # git commit -av --amend  # add current changes to previous commit
  # git reset --hard {branch|HEAD~n}  # revert to branch, or last n commits
  # git cherry-pick {branch|commit}    # copy specific patch to current branch
Committing Partial Changes

► Add individual patch hunks

```bash
# git add -p
diff --git a/obd_support.h b/obd_support.h
@@ -482,6 +482,8 @@ int obd_alloc_fail(const void *ptr,
    #define OBD_FAIL_LFSCK_NO_AUTO                    0x160b
    #define OBD_FAIL_LFSCK_NO_DOUBLESCAN       0x160c

+#define OBD_FAIL_FID_MAPPING                         0x1500
+
  /* Assign references to moved code to reduce code changes */
  #define OBD_FAIL_PRECHECK(id)       CFS_FAIL_PRECHECK(id)
  #define OBD_FAIL_CHECK(id)               CFS_FAIL_CHECK(id)

Stage this hunk [y,n,q,a,d,/,e,?]?
```
Rebasing Patches

- Patches should be kept up-to-date with master
- WC Checkpatch marks patches Verified: -1 that don't apply
- Multiple dependent patches in a single series

  # git rebase [-i] {master|b2_15}  # edit/reorder patches (interactively)
  # git add -u                     # add updated file(s) to commit
  # git rebase {--continue|--abort} # apply or discard changes
  # git rebase --skip              # drop current patch entirely
  # git rebase --onto {branch}     # rebase onto a new branch
Rebasing a Patch or Series

► Rebase last two patches

```bash
# git rebase -i {master|HEAD~2}
pick 4043c3c3bd LU-12885 llite: add enum ll_file_flags for clarity
fixup fc350ac3f4 LU-12885 mds: add enums for MDS_OPEN flags
reword 27723374a3 LU-16073 utils: double snapshot_mount fix
```

# Rebase fc350ac3f4.. 4043c3c3bd onto 27723374a3
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# These lines can be re-ordered; they are executed top to bottom.

► Can also change patch ordering in this way
(Re)basing Another Patch onto an Existing Change

# git fetch … & git checkout -b b_new_patch FETCH_HEAD
# checkout patch (#1, #2, #3)
# git cherry-pick {my patch}
# verify old commit hash is same (#4)
# apply new patch on top
# push new patch to Gerrit

► Do not rebase new branch onto latest master before submitting
  • Original developer should rebase or land their patch first

► Can also do simple rebase in Gerrit with [Rebase] button
  • "Change parent revision" and specify patch number
  • Cannot resolve patch conflicts directly in Gerrit
What To Do When Things Go Wrong

- All old commits/branches remain in the repository

```bash
# git rebase
3c96fb63e9 HEAD@{49}: rebase: LU-11721 tests: enable full debugging for sanity
da6cf33 HEAD@{1}: commit: LU-2468 libcfs: quiet debug messages
0b3e852 HEAD@{2}: checkout: moving from b_cleanup9 to b_cleanup7
5ea4d20 HEAD@{3}: rebase finished: returning to refs/heads/b_cleanup9
5ea4d20 HEAD@{4}: rebase: LU-1798 tests: don't set jobid if not changing
0b3e852 HEAD@{5}: checkout: moving from b_cleanup9 to 0b3e852a58^0
```

- If things go really badly, you can abort rebase and start again

```bash
# git rebase --abort
```
Extracting and Sharing Changes

Default values in $HOME/.gitconfig or .git/config in local repository

[user]
  name = Andreas Dilger
  email = andreas.dilger@whamcloud.com

[sendemail]
  smtpserver = shawmail.cg.shawcable.net
  smtpserverport = 25
  confirm = compose

[core]
  abbrev=12

[pretty]
  fixes=Fixes: %h ("%s")

[format]
  pretty=fuller
Thank You