Architecture Solution
MDT reply reconstruction improvement

version: 0.2
date: 2014/09/25
author: Grégoire Pichon - gregoire.pichon@bull.net

Introduction

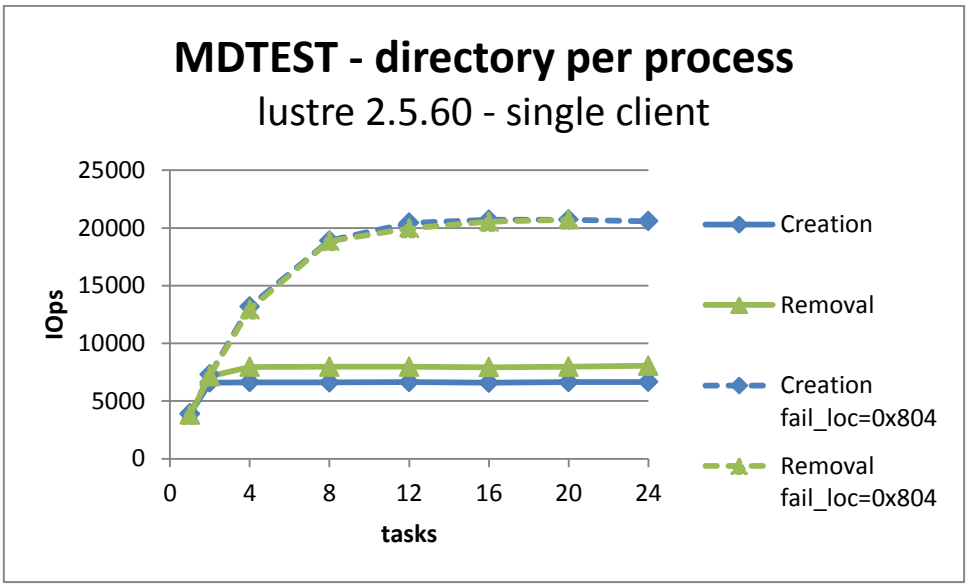
Currently, the MDT cannot handle more than one single filesystem-modifying RPC at a time, because there is only one slot per client in the MDT last_rcvd file. This slot is used to save the state of the last transaction (transaction number, xid, RPC result, operation data) so the reply can be reconstructed in case of RPC resend if the reply was lost. As a consequence, the filesystem-modifying MDC requests are serialized, leading to poor metadata performance from a single Lustre client.

The goal of this project is to support multiple slots per client for reply reconstruction of filesystem-modifying MDT requests, and to improve metadata operations performance of a single client.

This work is tracked with Lustre JIRA LU-5319.

Achievable performance

Here are the performance results obtained with *mdtest* benchmark on a single client with Lustre version 2.5.60. The filesystem is made of 1 ram-device MDT and 2 ram-device OSTs. Files are created with a stripecount of 1. The benchmark operates on 1 million files.



The plain lines show the file creation and removal rates on a single client in function of the number of tasks. The operation rates reach their maximum as soon as 4 tasks are running in parallel. This is because the Lustre client only allows one creation or removal request in flight at a time.

The dash lines show the file creation and removal rates with `fail_loc` value set to 0x804 on the client node. This error injection value actually allows to bypass the serialization of MDC requests, with the drawback that MDT reply reconstruction is not supported in that case (see Lustre JIRA LU-933). It gives an idea of the performance that could be reached when the feature described by this document is implemented.

Solution Requirements

Single client metadata performance

The performance of the filesystem-modifying metadata operations from a single client will be improved. A client will be able to have several modify MDT requests in flight.

Handling several MDT requests per client

The MDT will be able to handle several filesystem-modifying requests per client at the same time. Reply reconstruction feature will be enhanced to work in that context. Consistency of MDT operations and reply data on disk will be preserved.

Client - Server compatibility

Compatibility will be ensured between a client and a server that supports or does not support multiple modify MDT requests in parallel. The four possible configurations will be supported.

Upgrade

Upgrade and downgrade of Lustre client and server will be supported. However, MDS server downgrade will require all clients to be disconnected.

Solution Proposal

Connection

The MDT connection will allow specifying:

- a new flag that indicates support of multiple filesystem-modifying RPCs in parallel
- a new connection data that specifies the maximum number of modify RPCs in flight supported by the MDT

Client

A new mdc parameter will be available through the procfs directory `/proc/fs/lustre/mdc/*/` to define the maximum number of modify RPCs in flight allowed for a mdc. This will allow clients with different level of metadata activity, or different number of cores, to be able to send more or less modify metadata RPCs in parallel. The parameter value will have read-write access, but will be limited by the `max_rpcs_in_flight` value and the maximum number of modify RPCs in flight value returned by the MDT service during connection. Change of the parameter value will be dynamically taken into account. The client will ensure the number of modify metadata RPCs sent in parallel to the MDT will never exceed this value.

The existing code handles the close requests (`MDS_CLOSE` and `MDS_DONE_WRITING`) differently from other metadata requests. They benefit from a separate serialization allowing a close request to be sent in parallel of other requests. This avoids potential deadlock situations, where the processing of a modify metadata request triggers a lock cancellation, which requires a close request to be sent from the same client (see Lustre Bugzilla 3462). For the same reason, one additional RPC will be allowed for close requests above the maximum limit.

Server

The maximum number of modify metadata RPCs in flight supported by the MDT service will be tunable with a new MDT kernel module parameter. This parameter will have read-write access. Its value will be effective for new client connections.

The MDT reply data used for reconstruction is currently stored in the *last_rcvd* local file of the backing filesystem. This file contains an area for each client connected to the MDT. This file and its format will remain unchanged.

Another new local file will contain the reply data for the last modify RPCs of every connected clients.

Unit / Integration test plan

The single client metadata performance improvement will be verified with the *mdtest* benchmark. The benchmark will perform creation and unlink operations of a fixed total number of files, using a "directory per process" model.

MDT reply reconstruction and MDT recovery functionalities will be verified while performing several metadata operations in parallel from a single client. *auster* test suite will be enhanced with such test cases.

Upgrade, downgrade and inter-operability will be verified.

The *recovery-small* test suite will be run to verify no functional regression.

Acceptance Criteria

The "MDT reply reconstruction improvement" patch will be accepted as properly functioning if:

1. the single client metadata performance is improved
2. the MDT reply reconstruction and MDT recovery functionalities are verified
3. Upgrade, downgrade and inter-operability tests succeed
4. No functional regression is detected

Glossary

filesystem-modifying metadata operations : operations that modify the namespace of the file system, ie. file or directory creation, removal, rename, set attribute...
auster : test suite embedded in Lustre sources and packages
mdtest : metadata performance benchmark of the University of California