Interoperability - Client Recovery

Amit Sharma

22-Nov-2007

1 Introduction

The interoperability between the new 1.6.x clients and 1.8 server is an important task which will enable upgrading the existing 1.6.x clients to be able to talk to the 1.8 server which supports FID, and it requires that the 1.6.x client be made aware of the protocol that the 1.8 server follows. One of the areas where changes will be needed to support such interoperability is the client recovery.

This document will outline at a higher level design of what changes will be needed to the 1.6.x client to enable it to recover after any fault when it is running in the interoperability mode.

2 Requirements

- The recovery of the 1.6.x client (interoperable client) when talking to a 1.8 server should work seamlessly.
- The interoperable client should be developed on the 1.6 client.
- Interoperable client should be made to understand the protocol that 1.8 server understands, and also be able to communicate with a 1.6 server.

3 Client Recovery Process

We will review how the client recovery takes place, and also cite the differences in the process if any in 1.8 as compared to 1.6.

1. In 1.8 server the recovery task is handled by the recovery thread target_recovery_thread, which is started off by target_recovery_init(). This is different from 1.6 in which the recovery process is started as part of the target_handle_connect.

2. At the server end when the recovery process starts, a recovery_timer is started and the first transno to be replayed is found. And the recovery_timer is stopped after the time out period. If some client fails to connect within this time, it is evicted.

3. In the next stage, the clients replay their requests. If some client fails to replay its requests in the time out period it is evicted.

4. The next stage id for the clients to replay their locks. But if any client is unable to replay the locks in the time out period, the recovery is aborted and all clients are evicted. And the recovery process has to start again.

5. After this the server drops the recovering flag, and starts forwarding all the requests from now on to the regular mds_handle().

6. Then in the final stage, the server sends out final reply.

Most of the above is similar to 1.6, except that there are just two stages in 1.6 Replay and Final Reply. This will have some minor affect in the recovery process for the client in 1.6.x. And has to be taken care of by enabling the 1.6.x client to communicate with the 1.8 server in a way that the 1.8 client would behave in a similar situation.

4 Functional specification

From the point of view of the client, in case of recovery, most of the things would be the same except for the changes in the number and the format of the requests and replay messages that it sends to the server in the case of recovery as discussed above, and also due to the use of FIDs, we discuss some detail below. Although there have been many changes at the server end which affect the recovery process, mostly because of the new layered stack at the server end and some minor changes in the recovery algorithm. But the basic recovery algorithm from the point of view of the client doesnt change much and so most of those changes are transparent to the client and have very minor impact on the client's operation at the time of the recovery process.

To meet the above requirements, we need to make sure that the following differences in between the 1.6 and 1.8 are handled.

4.1 Use of FIDs

The use of FIDs brings about changes both at the client and the server end. These changes both at the server and client end will be taken care of as part of different tasks (info will be available in the hld of interop_server_fidea, interop_server_igif and client_interop_fid)

4.2 Request Flag and Connection Flag changes.

There have been few changes to the request formats from 1.6 to 1.8. New flags MSG_REQ_REPLAY_DONE and MSG_LOCK_REPLAY_DONE have been added and the flag MSG_AT_SUPPORT has been removed.

The Connection Flag also have changed. The flags OBD_CONNECT_CROW and OBD_CONNECT_TRANSNO are no longer used in 1.8. Although we are not quite sure these flags will be done away with or kept for the future.

New request flag MSG_CONNECT_TRANSNO has been added.

These changes need to be taken into account and the client needs to be made aware of these changes, so that it can talk to the 1.8 server in the interoperability mode.

As part of this task we would only be interested in the communication that happens in case of the client recovery, all other request/reply messages between the client and the server will be taken care of as part of the mixed_layout_req task. For more extensive detail on the request format changes the hld for client_interop_fids and client_interop_reqs can be referred.

4.3 SOM related changes

In case of client recovery, the client is evicted by the MDS and in case this client was the last writer for a file, then the MDS gets the replay information from the OST, and replays those steps in case there has been a attribute update. And then sends cancellation requests to the OSTs.

This recovery is transparent of other clients in the cluster, the MDS indicates them it does not have attribute caching enabled on such inodes and the clients obtain them from OSTs.

SOM related changes on client recovery are discussed in more details in the [som_recovery.lyx hld]

4.4 Algorithm changes from 1.6 to 1.8

As discussed in section 3, there have been some changes in the algorithm in the 1.8 as compared to 1.6, and this would mean the new 1.6.x client needs to respond differently to what it would have done when talking to the 1.6 server.

5 Use cases

5.1 The Client crashes/reboots

When the client comes up after a crash/reboot it will try to connect to the server. The server will find out if an export existed for this particular client and then start a recovery

process, as per the recovery algorithm.

5.2 The server crashes/reboots

In case the server crashes/reboots for some reason. When it comes up it would wait for a period of time allow clients to replay requests and locks. If a client is unable to replay its requests/locks during that time it will be evicted from the cluster.

5.3 Network failure

Network failure has to be handled in a similar way as the case when client crash/reboots. When the client requests time out, the recovery procedure will be kicked off and then the process follows as per the recovery algorithm

We can take a look at how a "open-write-close" scenario would work in case there is a network failure.

The network failure can happen at the following stages:

- Before the client does a "open": There could be two cases in open. One is the case of a pure open, and the other is the case of open/creat.
  - In case of pure open, the task is simple as a new trasaction is not created in this case, and the trans no is just bumped up. In case of failure such a transaction can be taken care of at the server end by a simple replay of the open operation.
  - In the open/create case there will be a transaction and so it will have to be handled. In the trasaction stop callback the transaction number, request id, last operation result and intent disposposition is stored in the last_rcvd record. At the time of replay this record will come in handy and since the file had already been created, it would just be opened as part of the recovery.
- After the "open" but before "write": There could again be two cases here, one is the simple case where in the clients request for open has been executed by the server so after the reconnect nothing needs to be done at the server end. The other case would be when the server did not receive the open request from the client. So, based on the status of the request on the client, it would start to replay requests, the server would compare the requests and see if that request has been executed at the server. And decide on whether to replay the transaction or not.
- Network failure after the "write" but before "close" and failure after "close" : The file open request (along with the fid for the newly created file) will be kept in the client replay list until the file is closed. There is an open file handle (struct mdt_file_data *mfd) on MDS for every open file, linked together into

this client's export. When client crash/reboot/reconnect to MDS, all open handle will be destroyed. When server crash/reboot/recover, client will replay its open request, and continue on the write operation.

5.4 Server upgrade

The server upgrade will be treated as a case of recovery from the point of view of the client. While upgrading the server to 1.8, it will be failed and for the clients it will be a normal recovery procedure. Just that in this case, they would be talking to a server or higher version after the recovery.

6 Logic specification

Below we see a brief of how the issues pointed out in the Functional Spec will be taken care of. More code level details will form a part of the DLD.

6.1 Use of FIDs

Most of the changes that are needed at the client side, from point of view of the use of FIDs will be taken care of as part of the interop_client_fid. But there will be some more changes that might be needed at the client side which might be very specific to the recovery task. Those will be taken care of as part of this task.

6.2 Request Format and Connection Flag changes

The interoperable client will be enabled to use the new request and connections flags when it is communicating with the 1.8 server, and to use the usual 1.6 communication protocol when communicating with the 1.6 server.

The recovery mechanism in 1.8 also works differently from the 1.6, this case can also be taken care by the client side changes by enabling it to communicate in a way which the 1.8 server understands. These changes will be worked out as part of the mixed_layout_req work. But there will be some client recovery specific messages/replies which will have to be taken care of as part of this task.

6.3 SOM Recovery

Most of the SOM recovery related work will be taken care as part of other task (som-recovery). It needs to be ascertained if any work will have to be done apart from that and will it fall in the scope of work for this task.

6.4 Algorithm changes

As mentioned above, there have been some changes from 1.6 to 1.8 from the point of view of the recovery algorithm, this can be handled at the client side by enabling the client to understand the changes in the algorithm and act accordingly.

7 Recovery changes

The whole task focuses on the recovery of the client in the interoperability mode.

8 Protocol changes

There will be changes needed to enable the 1.6.x talk to the 1.8 server in terms of the request format, although most of the communication will be taken care of as part of other task (client_interop_reqs), but some changes specific to recovery will have to be done as part of this task.

9 Focus for inspections

- Any other major difference in the 1.6 and 1.8 flags (connection, recovery etc.) that might have been missed and may have an impact on recovery particular and interoperability in general.
- Major protocol changes if any that have been missed in the new 1.8 server which may have an impact on the recovery of client.