whamcloud

Design Document
For the
Imperative Recovery Project
On the
ORNL Scalable File System Development Contract

Revision History

Date	Revision	Author
03/22/2011	Original	J. Xiong
03/28/2011	V0.99	J. Xiong
04/22/2011	V1.00	J. Xiong, eeb
06/27/2011	V1.01	J. Xiong
7/11/2011	V1.02	J. Xiong

I. Table of Contents

I. Introduction

The following solution architecture applies to the Imperative Recovery (IR) project within the ORNL Scalable File System Development contract signed 11/23/2010 and modified on 03/18/2011. ORNL is experiencing exceedingly long service outages during server failover and recovery. Currently this process takes unacceptably long because it depends on timeouts scaled to accommodate congested service latency.

Large-scale lustre have historically experienced problems recovering in a timely manner after a server failure. This is due to the way that clients detect the server failure and how the servers perform their recovery.

From the client side, the only way it can detect the failure of a server is by the timeout of Remote Procedure Calls (RPCs). If a server has crashed, the RPCs to the targets associated with that server will time out and the client will attempt to reconnect to the failed server, or to a failover server if configured. After reconnection, the client will participate in recovery and then continue in normal service.

A restarting target will try to recover its state by replaying RPCs executed but not committed by its previous incarnation. These RPCs must be executed in the original order to ensure the target reconstructs its state consistently. The target must therefore wait for all clients to reconnect before recovery completes and new requests can be serviced because a missed replay transactions could result in recovery failure and a late replay transaction could be invalidated by new requests. This wait, called the "recovery window" is bounded to ensure a failing client cannot delay resumption of service indefinitely.

The above processes are time consuming, as most are driven by the RPC timeout which must be scaled with system size to prevent false diagnosis of server death. This is especially difficult in an environment of ORNL's scale.

The purpose of imperative recovery is to reduce the recovery window by actively informing clients of server failure. The resulting reduction in the recovery window will minimize target downtime and therefore increase overall system availability.

II. Definitions

Target

A Target is a Lustre Object Storage Target (OST) or Metadata Target (MDT). A single server typically includes multiple targets.

Target Index

Target index is an integer to identify a target designated at formatting time or assigned by the MGS. Target index is immutable. For a target with name lustre-OSTXXXX, its index is XXXX.

Target Instance Number

A Target instance number is a unique number to identify and differentiate between each running instance of a target. The Target instance number is generated by the target itself when it starts up.

Target Status Table

A Target Status Table is a table on the MGS where every target's information is stored. The information in the target status table includes target identification information such as the file system name, target index and target instance number. It also includes target location information, namely the list of NIDs that can be used to reach this target.

Recovlock

The recovlock is a plain ldlm lock resident on the MGS, which is used by the MGS to notify the clients when the target status table was changed.

Recovery Window

The recovery window refers to the time from when a target is restarted to when it has completed recovery by reordering and replaying all uncommitted RPCs and is ready to serve new requests.

III. Requirements

Imperative Recovery will shorten recovery time by notifying clients actively when a server restarts.

Active notification means that the latency from server restart to clients starting to participate in recovery scales with message latency and not the RPC timeout.

Imperative Recovery will support failover servers

The notification delivered to clients on target restarts includes the NID of the server exporting the target. This ensures that when notified, clients connect immediately to the correct server both on server restart and on failover.

Imperative Recovery will only execute when all clients support Imperative Recovery

If the cluster consists of IR-supporting servers and non-IR clients, normal recovery must be adopted, to ensure non-IR clients participate in recovery.

Imperative Recovery will not impede the current recovery mode

Imperative recovery relies on the Management Server (MGS) to function properly. If the MGS is down, imperative recovery must not delay normal timeout-based recovery.

IV. Changes from Solution Architecture

The IR Solution Architecture proposed using llog infrastructure to transfer the target restarting information to the clients. Upon further analysis it was determined it would be more efficient for the MGS to maintain a target status table and use the GET_INFO RPC to transfer this information. This change does not impact any functionality described in the Solution Architecture.

V. Functional specification

In order to satisfy the requirements above, IR will actively notify clients after a target has restarted, allowing them to be reconnected promptly. The MGS was selected to provide this service, as all targets register with it after starting.

Target status table on the MGS

The target status table is used by the MGS to retain the status of each target in the cluster. Each target has a corresponding entry in the target status table containing the following information:

- File system name of target
- Target name
- Server index of target (i.e. - ost index or mdt index)
- Target instance number
- NID list by which the target can be reached
- Table version number when this entry was last updated

When a target is registered to the MGS, the MGS updates the target status table. If the target is a newly added one, the MGS composes a new table entry and adds it into the table; otherwise, for an already existing target, the table entry is updated, typically with the new NIDs and target instance number.

The target status table version is used to track changes to the table. Whenever the target status table is changed, this version number is incremented and new and modified entries are stamped with it. This makes it easy to find new or updated table entries since any previous version. The up-to-date table version number must be written to persistent storage so that it won't confuse clients in case the MGS itself is restarted.

The MGS will update the target status table while receiving target register RPC, and then notify a specified thread to enqueue an EXCL mode recover lock. The MGS has to respond target register RPC as soon as possible to avoid RPC timeout on the restarting target. If many targets register while the MGS is enqueueing or holding the recovlock, all their target status table entries can be changed in one go.

Clients use the recovlock to protect their cached copies of the target status table. On startup clients enqueue a shared recovlock and synchronize its version number with the MGS. When the MGS enqueues its EXCL lock, clients receive notification and must cancel and then re-enqueue their shared lock. On re-acquisition the target status table version number is used to find new and modified entries. Clients may then reconnect to new targets.

Note that clients may detect problems with a target, reconnect and recover independently of the MGS. This race is resolved by checking the target instance number when reading new target status table entries.

Target waiting policy

Unfortunately, it's impossible for the MGS to know how many clients have been successfully notified or whether a specific client has received the restarting target information. The only thing the MGS can do is tell the target that, for example, all clients are imperative recovery-capable, so it is not necessary to wait as long for all clients to reconnect. For this reason, we still require a timeout policy on the target side, but this timeout value can be much shorter than normal recovery. The exact time the target should wait

relies on the status of network, workload of previous instance and the size of cluster, therefore there will be a configurable entry under /proc to set this value. Typically the recommended value will be less than 5 minutes.

Since successful notification depends on all clients being registered with the MGS and clients cannot rely on notification to reconnect to the MGS after MGS failure, the MGS keeps IR disabled for a period on restarting.

Concurrent operation of normal and imperative recovery

Imperative recovery is not available when the MGS is down or some clients in the cluster do not support IR. IR therefore does not replace the traditional, timeout-based recovery but provides an additional mechanism that should accelerate recovery in most instances.

VI. Use Cases

Note: in the following use cases, when we refer to target shutdown, it means stopping the target by any means including administrative control, crashing, power loss or any other unspecified reasons.

Standard IR

Assumes an environment where the MGS has been running long enough for all clients to connect and no failover server is configured for the target. Shutdown a target, then restart it. Clients should be notified and then the same target will be reconnected; user should experience a much shorter interruption than during standard timeout-based recovery.

IR with Failover

Assumes an environment where the MGS has been running long enough for all clients to connect and failover pairs are configured. Shutdown the master server. Clients should be notified and then connect to the secondary server; user should experience a much shorter interruption than during standard timeout-based failover.

No IR – MGS Down

Assumes an environment where the MGS is down. Shutdown a target, then restart it. After restarting, target should wait for the normal timeout-based recovery window.

No IR - Not All Clients Support IR

Assumes an environment where the MGS has been running long enough that all clients have connected and the cluster is composed of a combination of IR-enabled clients and older clients that do not support IR. Shutdown a target, then restart it. After restarting, target should wait for normal timeout-based recovery window.

VII. Logic specification

This section describes how to use the target status table and recovlock to satsify the IR requirements.

Flow chart

Imperative Recovery Flow

Target status table

The target status table is updated whenever the MGS receives a target registration message. The MGS uses the target name and file system name as the key word to search the table. If there is no such entry in the table, it indicates that a new target has come online and a new entry will be composed in the table; otherwise, IR will update the table entry of that target (due to target instance number, it's impossible for the entry to remain unchaged after restarting), increase the table version number by 1, then enqueue an EXCL recover lock to notify the clients that the table has been changed.

The target status table entry for each target will include the following information:

- File system name of target
- Target name
- Server index of target (i.e. - ost index or mdt index)
- Target instance number
- NIDs by which the target can be reached
- Table version number when this entry was last updated

When a client is notified, it will fetch newly changed entries using a GET_INFO_RPC. It provides the MGS with the last table version number know to this client so that all new table entries created or modified by all restarting targets are retrieved.

After the clients receive the newly changed table entries, the target instance number in the entry will be used to match the instance number in the connection data. If they match, this indicates the client has already detected the restarting of that target; otherwise, a new connecting RPC will be issued.

A new config_llog_instance{} at client

Similar to config llog instance, a recover llog instance will be added on the client. The index of config_llog_instance is used to record the last-seen version number of the target status table. Whenever the clients' recovlock is revoked, this version number will be provided so the MGS knows which target's entries should be returned.

After fetching target status table entries, the client will reform those entries into lustre_cfg{} to facilitate code reuse. The lustre_cfg{} of target entry will have configuration strings in the following format:

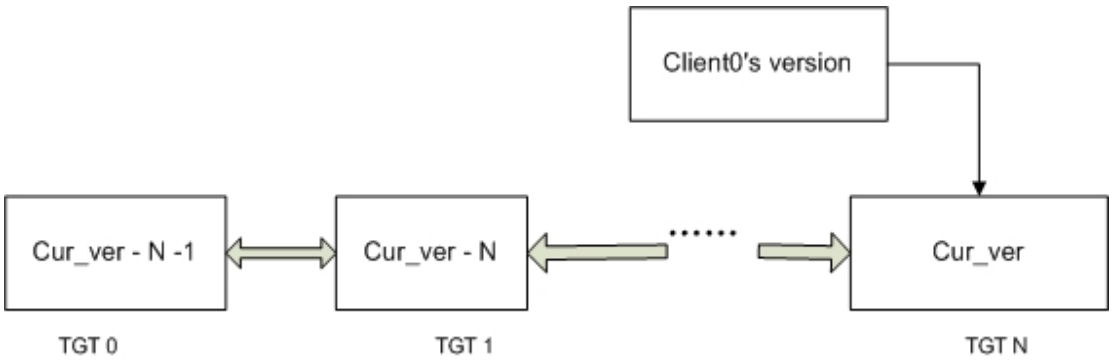
"osc.import = <NID>::<target instance number>"

For example:
"osc.import = 10.0.0.1@tcp::12345678" tells the client to connect to a new target with NID 10.0.0.1@tcp and that the target instance number is 12345678.

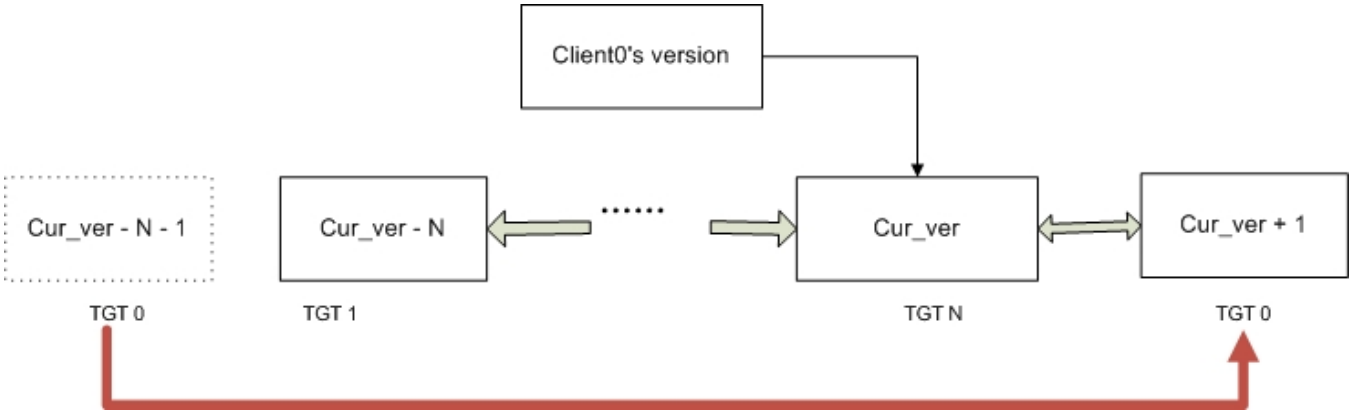
Target restart

This section describes what happens to the MGS target status table and what the clients will do after targets restart.

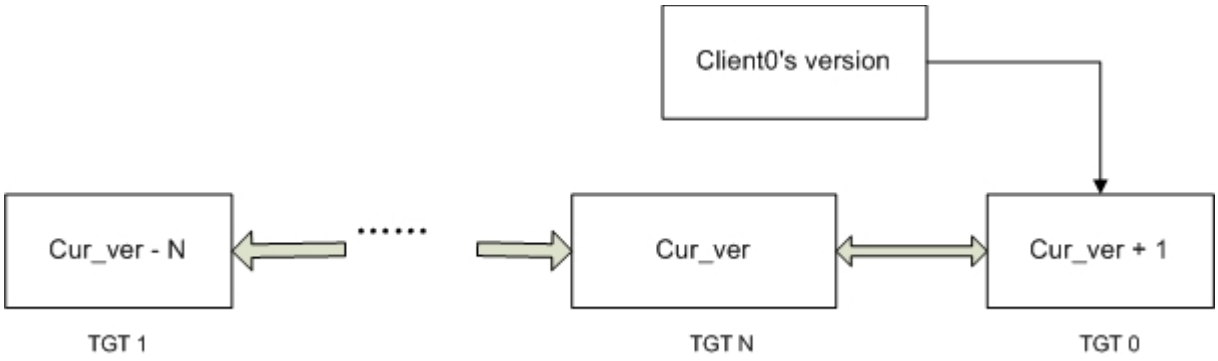
First of all, the target status table is managed as a linked list, sorted on increasing table version number.
At initial status, the version # of the table is Cur_ver:



Then assuming TGT 0 was restarted, the entry of TGT 0 will be updated by target register message, and then the entry will be moved into the tail of the list, and table version is changed to Cur_ver + 1:



Then, the MGS enqueues an EXCL mode of recovlock so clients will be notified. After re-acquiring the shared mode recovlock, clients will fetch target entries with version numbers newer than their last-seen version numbers. In this case, only TGT0's entry will be returned.
The final status will be:



This architecture supports multiple targets restarting at the same time.

MGS status and target policy

There are 4 different imperative recovery states on the MGS: **Partial**, **Startup**, **Disabled** and **Full**:

- **Partial** means not all clients support imperative recovery
- **Startup** means the MGS just started up but it's in a quiescent period when not all clients may be visible to the MGS
- **Full** means all clients are visible to the MGS, and all of them support imperative recovery
- **Disabled** is for test purpose. Imperative recovery will be disabled.

When a target is restarted, it will still use normal recovery as if there was no imperative recovery. Upon receiving the direction from the MGS in the

register reply RPC, targets will determine the appropriate time to close the recovery window. If the MGS informs a target to use imperative recovery, the time for the target to wait is a tunable parameter based upon specific site experience. The administrator should configure it based upon the size of cluster, delay on the network, workload, etc. For a medium size cluster, 5 minutes is typically long enough for all clients to reconnect to the restarted target.

Unless it is in the Full state, the MGS will inform the target to still use the standard recovery mode. Though the MGS may not be in the Full state, eligible clients still will be informed so that at least some of them can reconnect to the restarting target as soon as possible.

Connection flag

IR defines a new connection flag MSG_CONNECT_IMPERATIVE, which is used by the clients to tell the MGS they support imperative recovery. If the MGS supports imperative recovery (this is most cases as the MGS should be upgraded first), the same flag will be returned to the clients.

The MGS will maintain the IR-capability status of each file system. The MGS knows which file system the client belongs to when the client requests that file system's configuration.

On the MGS, whenever a new client connect or disconnect occurs, the MGS has to reevaluate the clients list to check if all remaining ones are now IR-capable. If the MGS is in Full/Reviving status, and a non-IR capable client is connected, the MGS status will be changed from Full/Startup to Partial; if the MGS is in Partial status, and if the last non-IR capable client is disconnected, the MGS status will be changed to Full status.

Configurable parameters

IR employs the following configurable parameters:

- Recovery_time_factor: this is a parameter on the target side representing the percentage of original target recovery timeout will be used. This parameter must be in the range of [10, 100]. If a target is instructed by the MGS that IR is usable, the recovery time is set to obd->obd_recovery_time * recovery_time_factor / 100. The default value of this parameter is 50. This parameter can be written into configuration logs.
- Imperative_recovery_status: this is a parameter on the MGS. Users may read IR status. The superuser may write the IR status for test purposes.

VIII. Wire protocol changes

- Connection flag MSG_CONNECT_IMPERATIVE
- Target instance number in obd_connect_data {} and mgs_target_info {}
- MGS_GET_INFO

```
struct req_format RQF_MGS_GET_INFO =
        DEFINE_REQ_FMT0("MGS_GET_INFO", mgs_get_info_client,
                        mgs_get_info_server);
struct req_msg_field *mgs_get_info_client[] = {
        &RMF_PTLRPC_BODY,
        &RMF_GETINFO_KEY,       /* Key: GET_TARGET_NID */
        &RMF_NAME,              /* File system name */
        &RMF_U64                /* Cached Table Version at the client */
};

struct req_msg_field *mgs_get_info_server[] = {
        &RMF_PTLRPC_BODY,
        &RMF_GETINFO_VAL,       /* Table Entries */
        &RMF_U64,       /* Client version of this RPC - CV;
                         * Client will set its cached version number to CV
                         * after processing this RPC
                         */
        &RMF_U64,       /* Latest Version at the MGS - LV;
                         * LV must be equal or greater than CV, and if
                         * LV > CV, client will read once again, until
                         * LV == CV
                         */
};

RMF_GETINFO_VAL contains an array of:
        struct mgs_recover_info {
                char            mri_tgtname[MTI_NAME_MAXLEN];
                __u32           mri_instance;
                __u32           mri_type;   /* LDD_F_SV_TYPE_OST or
                                             * LDD_F_SV_TYPE_MDT */
                __u32           mri_index;
                __u32           mri_nid_count;
                __u64           mri_nids[];
        };
```

IX. Open issues and future work

Configuration

It's a strong recommendation to separate the MGS and MDS for a large lustre deployment so that the MGS has enough bandwidth to handle IR workload. Also for a serious deployment always having master-slave MDS failover config, it will be interesting in running the MGS on the slave server. This will change the master-slave configuration to active-active similar to OSS failover nowadays.

Scalability

For simplicity, the MGS is used as a reflector to distribute events to clients. This may have scalability problem in cases where there are a large number of clients. This problem can be mitigated by separating the MDS and the MGS so that the workload on the MGS becomes lighter. However true scalability will only be achieved by distributing the task of client notification over the whole server cluster.

Usability

The MGS is a single point of failure – if the MGS is down, there is no way to know the failure of targets. The death of the MGS causes recovery to continue in standard mode. It is assumed this is acceptable.