Date:
2013-06-01

High Level Design – Client Health and Global Eviction

FOR EXTREME-SCALE COMPUTING RESEARCH AND DEVELOPMENT (FAST FORWARD) STORAGE AND I/O

MILESTONE: 4.1

LLNS Subcontract No.	B599860
Subcontractor Name	Intel Federal LLC
Subcontractor Address	2200 Mission College Blvd. Santa Clara, CA 95052

Table of Contents

Revision History

Date	Revision	Author
June 1, 2013	1.0	Isaac Huang

Use or disclosure of data contained on this sheet is subject to the restriction on the title page of this document.

Introduction

Currently Lustre PTLRPC requires a client to ping all its connected servers, as a mechanism for servers to monitor client health. This approach has two major problems:

- Protocol overhead grows linearly with the product of number of clients and number of servers. In extreme cases, ping storms have been observed which hogged down networks and LNet routers.
- Servers make individual decisions as to when to evict a client. When a client crashes, the server it has pinged most recently will take the longest time to evict the client, despite that other servers may have evicted it already.

Now with the help of scalable server collectives and server discovery service, we can address it without introducing drastic protocol changes. The core ideas are:

- A client chooses a master server and pings only this server.
- The master server keeps track of client health, and when necessary performs a global eviction by server collectives.
- When server discovery module declares the master server as dead, other connected servers wait for the client to choose a new master before they evict the client locally.

In contrast with the current approach:

- Protocol overhead can be reduced by a factor of the number of servers.
- Servers all evict a client at a same time. Only the master server makes decisions to evict globally.

Definitions

The following definitions are used throughout the rest of this document:

- Master server: The server which is in charge of monitoring a client's health. It's chosen by the client.
- Server quorum: A server considers itself to be part of the quorum when the total number of live servers exceeds half of the total number of known servers. Otherwise it's said to be absent from quorum. This information is provided by the server discovery module.
- Connection: A Lustre PTLRPC connection between a client and a server. It usually involves connection handshake to exchange capability flags, creating memory states like exports and imports, and creating server-side persistent state like a new entry in the last_rcvd file.

Changes from Solution Architecture

Specification

Here’s a brief outline of the protocol:

1. Client picks a master server at random, and must establish connection to it before the client tries to connect to any other server.
2. Client tells each server (including its master server) it connects to who his master server is. Servers remember the master server of each connected client as long as the connection is active. Master server keeps track of client health by keeping a simple last-seen timestamp, updated every time a message is received from the client.
3. Client keeps pinging its master server at regular interval, unless it has exchanged other messages with the server recently.
4. When a master server notice a client is dead, i.e. the client’s last-seen timestamp is too old, it broadcasts to all servers saying “evict this client”. Note that the master server doesn’t know which servers the client has connected to, so it must broadcast to all live servers.
5. When a server receives an eviction request from server A:
    a. If the client is not connected, simply reply with “client not connected”. Otherwise proceed to the next step.
    b. If the client’s master server is not server A, reply with “client’s master is server B”. Otherwise proceed to the next step.
    c. Evict the client, and reply with “client evicted”.
6. Master server receives aggregated replies to its eviction request, and examine each server’s response:
    a. Server replied “client evicted”: good.
    b. Server replied “client not connected”: this is expected.
    c. Server replied “client’s master is server B”: ignore.
    d. Tear down client connection and stop monitoring its health.
7. If a client hasn’t received any message from its master server after a timeout, client chooses a new master from currently connected servers by sending him a message “you are my new master”. Server replies with success. If server doesn’t reply in time, repeat a new master has responded. Then client sends a notification to the rest of the connected servers, saying “my new master is B”.
8. When a server notices that a connected client’s master server has died, by notification from server discovery module, it puts the client into orphaned state:
    a. If the client sends an update on his new master in time, update the record, moves client back to normal state. Otherwise:
    b. Evict the client locally.

Protocol details are given in the following sections for servers and clients. Note that a server can be both a master server and a non-master server for different clients. The following sections describe the logical roles of master server and non-master server separately for clarity.

Master Server

Master server handles the following protocol events:

- Connection request from clients
- Monitor client health and evict clients
- Incoming eviction requests
- Absence of server quorum.

Handling Connection Request

When a client connects to its master server, the server:

1. Reject the connection request if it’s absent from server quorum.
2. If it’s already connected to the client, which can happen when a client quickly reboots, the previous connection states must be cleaned up:
    a. If the server is not the client’s previous master, tear down previous connection, accept the connection request and mention “Your previous master was A”
    b. If the server also happens to be the client’s previous master server, it issues a global eviction request and waits for the eviction to complete.
3. Accept the current connection request and reply with success.

Note that the global eviction from step 2.b will only tear down connections made by the client’s previous incarnation. The client is still connecting to a new master, so it couldn’t have established any connection yet.

Client Timeout and Eviction

Master server keeps track of slave client health by keeping a simple last-seen timestamp, updated every time a message is received from the client. If this timestamp hasn’t updated after a timeout, the server issues a global eviction.

The server can batch several evictions in a single collective request, e.g. by queuing upcoming evictions and processing the queue once per second.

A client can be considered as globally evicted only once all servers have been notified. The master server should keep retrying until all servers are notified.

Incoming Eviction Request

Ignore the request and reply “Client’s master is actually me”. The eviction could have come from the client’s previous master, e.g. the client has rebooted and chosen me to be its new master.

Absence of Server Quorum

This can happen when there's a partition in the server network, and a master server falls in the minority part of the partition. The master server should evict all its clients locally, as a global eviction wouldn't be able to reach the majority of the servers. As for non-master servers for the master's clients:

- If they are in the quorum (i.e. the majority part of a network partition), server discovery module will declare the master server as dead and they will put the clients in orphaned state, and eventually evict the client unless the client has chosen a new master server.
- If they're not in the quorum, they will evict the clients locally.

In both cases, the client will be evicted globally. Please also see Risks & Unknowns.

Non-master Server

- Connection request from clients
- Incoming eviction requests
- Absence of server quorum.

Handling Connection Request

When a client connects to its non-master server, the server:

1. Reject the connection request if it's absent from server quorum.
2. If it's already connected to the client, which can happen when a client quickly reboots before its previous master evicts it, the previous connection states must be cleaned up:
    a. If the client is in orphaned state, tear down the old connection and jump to step 3.
    b. If the server is not the client's previous master, tear down old connection and accept the request with the message "Your previous master was A"
    c. If the server also happens to be the client's previous master server, it issues a global eviction request and waits for the eviction to complete.
3. If there's no existing connection with the client, and the client's master is dead, reject request with reason "Master already dead".
4. Accept the current connection request, keep record of the client's master server, and reply with success.

Note that the global eviction from step 2.c will only tear down connections made by the client's previous incarnation. To reach this point, the client must have established connection with a new master server which is different from this server, so all the newly connected servers will ignore the 2.c eviction request as it comes from the old master server.

In addition, server trusts the client's claim that a master server has agreed to monitor its health. If the client lies about it, then it will never get evicted, since there's no master server keeping an eye on its health. If necessary it can be solved by:

1. The master server returns a signed signature to a client when accepting its connection request.

2. Client presents this signature in the connection request to non-master servers.
3. Non-master servers must validate the signature.

Incoming Eviction Request

If the eviction request comes from the client’s current master server, then evict the client.

Otherwise, reply “client’s master is not you”. This situation can happen when a client has rebooted and connected to a set of new servers and then its previous master server tries to evict it.

Absence of Server Quorum

This can happen when there’s a partition in server network. Evict all clients locally.

Clients

Clients perform the following protocol operations:

- Choose a master server and connect to it
- Ping master server at regular interval
- Handle evictions.

Connect to master server

Client picks a mater server at random, and must establish connection to it before the client tries to connect to any other server. Client must pick another master if it fails to connect to the currently chosen master, and repeats this process until a connection has been created to a master server.

Once a master server is connected, normally the client doesn’t change its master, but on the following occasions, a client must choose a new master server (preferably from its connected servers) and notifies all currently connected servers about this new choice:

- Master server doesn’t respond.
- Master server disconnects, e.g. it has become absent from server quorum.
- Non-master server rejects connection request with reason “Master already dead”.

If the chosen server accepts with message “Your previous master is A”, there’s previous connections that needs to be evicted. Then the client tries to connect to server A as a non-master server:

- If server A receives the connection request, it’ll issue a global eviction to clean up all previous connections and then accepts the connection request.
- If server A has died, non-master servers will orphan the client’s previous connections and eventually evict them locally.
- If server A is not reachable, it will evict the client globally as the client no longer pings it.

Ping Master Server

Client keeps pinging its master server at regular interval, unless it has exchanged other messages with the server recently. In other words, a client makes sure it sends at least one message to the master server during a predefined ping interval.

The ping interval must be shorter than server eviction timeout.

Connect to non-master server

The client must have established connection with a master server at this point. If the master server is aware of any previous connection states, the connections must have been evicted already. But in the case the master wasn't connected to the client previously, it could happen that a non-master server can accept the client's connection request with message "Your previous master was A". Then the client tries to connect to old master server A as a non-master server in an attempt to evict all previous connections:

- If server A receives the connection request, it'll issue a global eviction to clean up all previous connections and then accepts the connection request.
- If server A has died, non-master servers will orphan the client's previous connections and eventually evict them locally.
- If server A is not reachable from the client, it will evict the client globally as the client no longer pings it.

The connection request can also be rejected with reason "Master already dead". This happens when the client's master service has died but the client hasn't noticed it yet. Now the client must connect to a new master server and notify all currently connected non-master servers about this new choice.

Evictions

A client should evict all its local imports as soon as it learns from one server that it has been globally evicted.

API and Protocol Additions and Changes

Open Issues

Describe any open issues that remain to be answered at this time.

Risks & Unknowns

In the current design, if a server network partition happens in a way that divides servers into two groups yet both groups still have good connectivity to the clients, servers not in the quorum will evict their clients despite that clients have good connections to them. As client evictions can result in data loss to the end user, we're exploring ways to handle this situation without evictions and without adding too much complexity to the protocol.