CLIO Simplification Scope Statement

Signed-off
This document was signed-off by OpenSFS on 16th November 2013.

Introduction

The following scope statement applies to the CLIO Simplification Design project within the Technical Proposal by High Performance Data Division of Intel for OpenSFS Contract SFS-DEV-003 signed Friday 23rd August, 2013.

Problem Statement

The Lustre* file system client implementation for the IO path (called CLIO) is responsible for issuing RPC commands for reading and writing data to the OSTs. CLIO was reconstructed in Lustre software version 2.0 for cross-platform portability. The 2.x client implementation has proven over time to be highly complex thus making the code hard to understand and maintain.

Liblustre has not been maintained or tested for a number of release cycles. Today, the code compiles successfully but no evidence to show that anybody is using liblustre. This code only serves to confuse engineers and it should be removed.

In addition, CLIO is highly complex, with a significant amount of this complexity being contributed by the cl_lock. This complexity creates a high barrier to entry for engineers wanting to modify, enhance and identify bugs. As a result enhancements to the client code are time consuming and a significant number of BUGs arrive from the cl_lock portions of the code. These BUGs can only be fixed by a few engineers because of the code complexity. LLNL has mentioned that "even our experienced engineers can't understand the Client IO code". The most complex part of cl_lock should be removed to reduce complexity and confusion. After this is complete CLIO will be easier to understand.

Project Goals

- Deliver a solution architecture.
- Deliver a high-level design.
- Simplify client side design.
- Identify unused features for removal.

In Scope

1. Analyze and design a simplified implementation of cl_lock by implementing a cache-less lock. This reduces code complexity and has been a source of bugs in the past.
2. Review and identify opportunities to improve file ioctl() calls for better integration into the CLIO code. This will allow removal of old OBD API infrastructure that is only used by this code. These file ioctl() calls are obd_find_cbdata(), ll_lov_getstripe(), ll_do_fiemap(), and ll_data_version().
3. Design the removal of obsolete OBD API callbacks such as obd_brw(), obd_punch(), etc. The dead code misleads and confuses developers not familiar with the code, and makes it more difficult to understand what APIs are actually in use.
4. Analysis of removing liblustre, WinNT and MacOS code and layering. The liblustre code is not used by Lustre today has not been tested in several years. There are ongoing costs to maintain the abstractions needed by the non-Linux platforms, due to the different VM/VFS abstractions compared to Linux so there is a maintenance and complexity cost without any clear benefit.

Out of Scope

- Code will not be implemented as part of this project.
- This project is not specifically targeted at improving performance. There should not be any negative performance impact from these changes, though performance improvements are of course welcome.

Project Constraints

Bobi Jam and Jinshan Xiong are the strongly preferred engineers for this project.

Design should take the Data on MDT and File-Level Replication Design project constraints into account.

Key Deliverables

- Solution Architecture.
- High-level design document.
- Implementation assessment.
