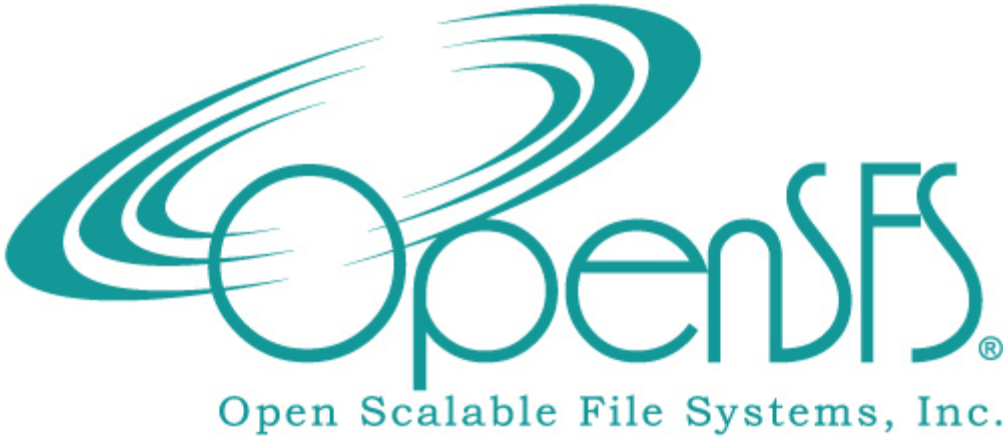

OpenSFS

Lustre feature development

Technical Specification

Version 1.0

OpenSFS Solicitation W4570

February 20, 2013

Background

The work described herein is in support of foundational changes to the Lustre file system, which will improve Lustre in four broad categories:

1. File system availability and robustness
2. Storage management
3. Performance
4. Lustre networking

The OpenSFS Technical Working Group (TWG) recently gathered requirements from the Lustre community for new Lustre features. A summary of the results of this exercise is available online for further reference[1]. This Technical Specification briefly describes the four areas of improvement identified by the TWG as necessary to provide a strong foundation for future Lustre roadmap development. OpenSFS recognizes that these features are very important for both our participants and the broader Lustre community.

OpenSFS is seeking proposals to develop designs, cost estimates, and project plans for improvement in these and other areas that Offeror deems appropriate. It is important to note that OpenSFS is not seeking proposals that include the full scoping of work required to implement features in Lustre. Instead, OpenSFS is seeking proposals that will provide sufficient detail to justify detailed scoping and design by the Offeror. The result of this investigation will be for the Offeror to provide a detailed development proposal. OpenSFS may or may not subsequently fund the full development of features proposed by the Offeror.

Offeror may propose scoping and design activities for any or all elements identified in the four broad categories described in this Specification or for other elements that the Offeror deems relevant.

1.0 File system availability and robustness

Availability encompasses requirements to make Lustre more robust and better able to tolerate errors. The highest priority by consensus is to address Lustre's dependence on timeouts and requires development to avoid timeouts as is best possible. This category also requires improved fault management

The following requirements address improvements to Lustre availability, fault tolerance and recovery at future system scales. Investment in one of these technologies now provides the foundation that Lustre needs to achieve the next levels of system scale.

[1] http://wiki.opensfs.org/images/f/f9/OpenSFSTWGRequirements2012.pdf

Offerors should note that there are funded efforts through the DOE FastForward program[2] for exascale system research to prototype scalable fault detection and resiliency mechanisms. There also exist ongoing scalability challenges that could be addressed in the near term[3].

Scalable fault management

While it is already the case that today's supercomputers have a marked dependence on their file systems for productive use, this dependency will continue to rise as we see more and more center-wide file systems. Today, large-scale deployments of Lustre may require tens of minutes to recover from faults. To minimize the downtime for the entire center, reliability must increase and recovery from faults must be substantially faster than today.

Lustre must expose errors it detects to standard administrative infrastructures. We cannot continue with error logs as being used today. Instead, Lustre must detect, collect, and parse faults then distribute the errors in a scalable manner to the administrative interface for notification.

Avoid RPC timeouts

Users sometimes perceive Lustre as unstable because of periodic pauses in execution as Lustre waits for an overloaded server, or a timeout to expire. We have discussed health networks as a means of providing lower latency fault detection and improved error handling. This will be a scalability feature that by providing an active, deterministic mechanism for communicating system status will avoid the sequence of cascading timeouts that limits Lustre at scale. For example, the current use of pings is difficult to tune at scale, and also imposes a significant overhead on the network, impact to IO performance, and impact to OS noise/jitter. The health network should be a high priority, efficient and reliable communications channel to avoid the need for timeouts and make client/server interactions more deterministic. As a result, we expect work fulfilling this requirement to improve recovery times, facilitate error detection within Lustre and improve file system responsiveness.

2.0 Storage management

Storage Management is a new area for Lustre that builds on the foundational work started by CEA for their Hierarchical Storage Management (HSM) project. In this category, OpenSFS seeks to extend CEA's work to make Lustre more relevant in modern data centers and more competitive with other file systems by enabling enterprise class features such as object migration, file mirroring, and replication. These changes will require a common infrastructure, on which creating these closely related features and enhancements would be easily possible.

[2] https://asc.llnl.gov/fastforward/
[3] For example, see http://jira.whamcloud.com/browse/LU-7.

HSM and storage management infrastructure

There is an ongoing project to implement HSM for Lustre that provides the foundational infrastructure needed for migrating data inside Lustre or among external storage systems. The current implementation of HSM, which pushes updates from the Lustre change log into a policy engine database, is inefficient, especially under high metadata load. Enabling object migration in Lustre will require a more scalable change log and layout lock design that provides consistent behavior and ensures policy engines utilizing this feature do not negatively impact performance.

Offerors should note that work related to CEA's HSM to improve file and object layouts within Lustre (multiple layout support, related to extent) is currently funded through the point2bdmc ITEA project[4].

OST migration/rebalancing

Object migration and OST rebalancing are examples of functionality long-missing from Lustre that the above enhancements will facilitate. We seek a utility to move Lustre objects between OSTs to more evenly distribute free space among the OSTs or to distribute objects to new OSTs added to expand the file system. This same facility can be used to manage space usage between tiers (OST pools) of storage to allow configurations with burst buffers, and archival disks. Similarly, it should be possible to migrate all objects off of one or more OSTs before they are replaced or removed from the file system. Such a facility must ensure that current workloads are not overly impacted by this activity.

3.0 Performance

Performance continues to be an area of concern. OpenSFS is hopeful that the new metadata features currently under development (SMP affinity, distributed namespace) will greatly improve performance of the metadata server. Nonetheless, there are workloads where Lustre performance continues to need improvement and we want to address architectural bottlenecks for both bulk I/O and metadata performance of a single Lustre client.

The requirements in this section highlight areas where Lustre performance could be improved. Deliverables that meet these requirements should provide immediate benefits.

Single client performance

Single client performance (CLIO) under Lustre 2 has degraded from Lustre 1.8. Although new multi-threaded RPC code has improved performance of a single-threaded reader/writer, there are still bottlenecks, such as the number of simultaneous RPCs in flight, internal lock contention, and SMP-unfriendly code, that prevent a single client from maximizing the performance available from the Lustre file system and its interconnect network. The client is

[4] http://www.itea2.org/project/index/view/?project=10184

currently also limited to a single metadata-modifying RPC in flight, which will also impact DNE MDT performance. Solutions that improve performance of both bulk I/O and metadata operations are desired.

File create performance

Previous work to improve metadata performance on a single MDS have gained significant improvements to directory and device node creation, but the benefit to creating files with objects on the OSTs has been less clear. Thus, file creation performance (in particular, OST object precreation) remains an area where Lustre requires significant effort to meet user requirements[5]. As with the MDS performance tuning, there are likely also SMP scaling bottlenecks in the OST code that can be addressed as part of a larger performance review.

Directory traversal and attribute retrieval

Directory listings performance has increased with recent metadata projects, but "ls -l" speeds for a single client should still be improved. Some possible areas for exploration are client-OST interactions or some version of a size-on-mds mechanism (e.g. synchronous recording of open-for-write, using the HSM "dirty" flag, or simplifications for single-client access).

4.0 Lustre networking

Lustre networking (LNET) is the transport for remote procedure calls (RPCs) to the Lustre servers. OpenSFS has identified a number of requirements for improving scalability, configurability and reliability of Lustre by enhancing core networking functionality.

As more compute systems use shared Lustre file systems, the robustness and configuration of the LNET layer will become more critical to successful file system deployments.

LNET channel bonding

LNET routers and servers are currently limited to a single channel provided by a single instance of the LND. This restriction limits bandwidth and reliability of an LNET connection to a single interconnect. LNET should allow multiple LNDs to be bonded as a group in order to enable load balancing and failover between LNET endpoints.

Improved LNET robustness

The original LNET design used multiple routers to guarantee connectivity, but performance suffers when there are large numbers of routers. This effect can be exacerbated when there are multiple network levels as router transmit credits become depleted within a network. Furthermore, performance of a group of routers can suffer by one poorly behaving router. Proposed work should consider mechanisms for improving LNET robustness and router performance.

[5] See the first appendix of the OpenSFS TWG Requirements report: http://wiki.opensfs.org/images/f/f9/OpenSFSTWGRequirements2012.pdf

Dynamic LNET configuration

LNET configuration currently uses static routes and requires LNET to restart to capture configuration changes. LNET needs to adopt a more IP-like configuration so that network changes can be more easily programmed and qualified.