SMP Node Affinity Scope Statement

Introduction

The following scope statement applies to the SMP Node Affinity Scope Statement project within the SFS-DEV-001 contract/SOW dates 08/01/2011.

Problem Statement

Followers of CPU design are observing steady increases in core count. As core count increases thread resource contention (particularly locking) threatens to throttle Lustre server performance.

A multi-core CPUs are now common. Core count is expected to continue climbing and as it does a Lustre server will need to increase thread count to exploit the CPU capacity. On Lustre today additional threads will contend a single wait-queue, share one request queue, and share only a couple of global locks. Hence, without additional work, Lustre servers are expected to perform badly on multi-core CPUs for reasons including:

- Overhead of lock contention.
- Overhead of process switching between cores.
- Scheduler attempting to balance threads across cores.

These factors can be addressed by splitting the computing cores into configurable Compute Partitions. Lustre RPC service threads will then be bound to a specific Compute Partition. This design:

- Reduces the overhead of threads switching cores by keeping the thread running on the same core as the cache memory.
- Avoids needless contention on the inter-CPU memory subsystem.
- Keeps RPC request processing local to the resources that they affect.
- Allows the protocol stack to scale effectively as the number of cores increases.

Project Goals

SMP node affinity is concerned with improving vertical scalability of a Lustre server by addressing software insufficiency on multi-core machines. This will enable Lustre to fully exploit increasingly powerful server hardware as it becomes available.

SMP node affinity will implement the following features:

- General libcfs APIs to provide a framework to support Compute Partitions
- Fine-grained SMP locking for Lustre
- SMP node affinity threading mode for Lustre

Benchmarks for metadata performance will be made on a cluster with a reasonable number of clients and a large (12+) number of cores.

The new features will demonstrate that performance of a single metadata server with a large (12+) cores is improved for file operations: create/remove/stat.

Code will land on WC-Lustre master branch.

In-Scope

- High level design document for SMP node affinity of Lustre.
- Improve small message rate of LNET on multiple (12+) core server.

- Improve small RPC rate of ptlrpc on multiple (12+) core server.
- Improve general metadata performance on a multiple (12+) core server for narrow strip-count file.
- OST stack performance will not exhibit regression.
- SMP node affinity code will take place against WC-Lustre 2.x baseline. Code will cover:
    - General SMP improvements for Lustre and LNET.
    - libcfs APIs to support CPU partition and NUMA allocators.
    - SMP node affinity threading mode for Lustre and LNET.
        - LND threads
        - ptlrpc service threads
        - ptlrpc reply handling threads
- Update manual to include SMP node affinity tuning parameters.

Out of Scope

- Wide strip count files involve additional factors that will not be helpful for demonstrating performance improvements.

Project Constraints

- Liang is the only engineer with the expertise to lead this work.
- Test cluster with suitable multiple (12+) core nodes available to WC engineers regardless of nationality.

Key Deliverables

- Signed-off Milestone document for the project phases:
    - Solution Architecture
    - High-level Design
- Test plan
- Source code against WC-Lustre 2.x that implements the feature requirements and runs on the test cluster.
- Code landed on Master WC-Lustre 2.x as Issue LU-56.

Glossary

- CPU partition
  CPU partition is subset of processing resource of system, a CPU partition could be any number of CPU cores in system: it can be a single core, or any specified number of cores, or stand for all cores in a NUMA node, or represent all CPUs of a system. The number of CPU partitions can be set by libcfs APIs.
  A fat server will be divided into several compute partitions, each compute partition contains: cores in CPU-partition, memory pool, message queue, threads pool, it's a little like virtual machine, although it's much simpler since the the Lustre RPC processing threads will still have access to all Lustre global state if needed.