Demonstration Milestone for Parallel Directory Operations

This milestone was submitted to the PAC for review on 2012-03-23. This document was signed off on 2012-04-06.

Overview

This document describes the work required to demonstrate that the Parallel Directory Operations code meets the agreed acceptability criteria. The Parallel Directory Operations code is functionally complete. The purpose of this final demonstration phase is to show that the code provides an enhancement to Lustre when used in a production-like environment.

Acceptance Criteria

PDO patch will be accepted as properly functioning if:

1. Improved parallel create / remove performance under large shared directory for mknod and narrow stripe files (0 – 4) on multi-core server (8+ cores)
2. No performance regression for other metadata performance tests
3. No functionality regression

Baseline

The baseline measurement for this Demonstration will be Lustre 2.1 enhanced with Parallel Directory Operations code. Baseline measurements will be made with the Parallel Directory Operations disabled.

Parallel Directory Operations and related performance enhancements has been included in Lustre-Master since build #431

The isolated Parallel Directory Operations code can be found as commit 8e85867.

NOTE: In order to deliver the maximum benefit of the Parallel Directory Operations work a number of additional enhancements were made to Lustre. These include

- Multiple OI tables (LU-822)
- Configurable BH LRU size (LU-50)
- Root node BH ref of IAM container (LU-32)

These enhancements remain active even with Parallel Directory Operations is disabled. As a result, the overall performance increase observed simply by enabling Parallel Directory Operations is reduced compared to a Lustre version that predates the beginning of the Parallel Directory Operations work.

Hardware platform

Demonstration will take place on the Whamcloud cluster Toro. The hardware specification include:

- 16 Client Nodes, 3 Server Nodes (1 x MDS, 2 x OSS)
- MDS node are of type 'Fat Intel Node'
  - 2 x Intel Xeon(R) X5650 2.67GHz Six-core Processor (2-HT each core)
  - 16GB DDR3 1333MHz Memory
  - 2PT 40Gb/s 4X QSFP InfiniBand adapter card (Mellanox MT26428)

    - 1 QDR IB port on motherboard
    - SSD as external journal device (INTEL SSDSA2CW120G3), SATA II Enterprise Hard Drive as MDT (single disk, WDC WD2502ABYS-02B7A0),
- OSS Nodes are of type: 'Fat AMD Node'
    - 2 x AMD Opteron 6128 2.0GHz Eight-Core Processor
    - 16GB DDR3 1333MHz Memory
    - 2PT 40Gb/s 4X QSFP InfiniBand adapter card (Mellanox MT26428)
    - 1 QDR IB port on motherboard
    - 3 x 1TB SATA II Enterprise Hard Drive (single disk, WDC WD1003FBYX-01Y7B0)
- Client Nodes are of type: 'Type 1 Thin Clients'
- 2 x Quad-Core Intel E5507 2.26G/4MB/800
    - Mellanox ConnectX 6 QDR Infiniband 40Gbps Controller (MTS3600Q-1BNC)
    - 12GB DDR3 1333MHz E/R memory
    - 3.5" 250GB SATA II RAID Enterprise Hard Drive (SATA II 3.0Gb/s, 16MB Buffer, 7200 RPM)
- Infiniband between all the nodes: MTS3600Q managed QDR switch with 36 ports.

Test Methodology

The intention of this phase of testing is to demonstrate the Parallel Directory Operations behaves in accordance with the acceptance criteria within a realistic production-like environment (defined above). To perform a test in this environment, the following tools are used:

mds_survey

`mds_survey` is a metadata performance benchmarking tool developed by Whamcloud, it can only be used with the Lustre software stack. Current version of `mds_survey` will run over MDD directly which will bypass RPC and LDLM layers, which means it can show pure metadata performance of MDS local stack. User can specify 1-N threads to create/stat/remove files under 1-M target directories. Because goal of this project is improving shared directory metadata performance tests are carried out with a single target directory.

Results gained from `mds_survey` are attached to LU-822.

mdtest

mdtest is used for testing the metadata performance of a filesystem, we use it measure parallel directory operations performance with whole Lustre stack, also we can verify whether there is any performance drop for single thread metadata performance. We will use mpirun to launch mdtest on client nodes so they can parallel operate under target directory, here is a short example of using mdtest:

```
mpirun \--machinefile mfile.16 \-np 512 mdtest \-d /mnt/lustre/testdir \-n 2048
&amp;nbsp;-F \-C \-r
```

`mfile.16` is a regular text file that contains a list of client hostnames. `-np 512` indicate that a total of 512 mdtest threads will be created on the client nodes, `-n 2048` means that each thread will create/remove 2048 files under target directory.

Oprofile

In previous round of tests, it was realized that `oprofile` would not help because significant issues remain with

uncomplete SMP locking work. Without this work, `oprofile` only exposes locking contention overhead. For this reason, `oprofile` data is omitted from this test.

Test filesystem configuration.

- 16 Clients, each with 16 threads.
- Test with shared directory and with unique directory to show no regression.
- mknod, opencreate (0-stripe), opencreate (1-stripe), opencreate (2-stripe), opencreate (4-stripe), unlink
- Time the duration to repeat operations one million times.
- Test repeated five times. Median and max are recorded.

Test results

The following assets will be collected from each test:

- mds_survey output
- mdtest output
- Oprofile output

Test duration

Each test iteration is expected to take two hours. Each test requires five iterations and there are five different tests to complete. The expected duration of the test is in the order of three days uninterrupted operation.

Appendix A

Benchmark characteristics

mknod

One single rpc is issued.

Open-create

Open-create benchmarks also include a close command. This means two rpc's are required.

Appendix B

Path-finding Demonstration results from December 2011.

whamcloud

Optional Where
Optional Date

Parallel Directory Operations performance tests

- Liang Zhen
  Whamcloud, Inc.
  liang@whamcloud.com

whamcloud

Testing environment

- MDS
  - 2 Intel 5650 6-core CPUs (2-HT each core)
    - Total 12 cores, 24 HTs
  - 24G RAM
  - 1 SSD external journal, Sata disk
- OSS
  - 16G RAM
  - 2 OSSs, 2 OSTs on each OSS
- 16 clients
  - Intel Xeon E5507: 4-core
- Networking
  - QDR infiniband

whamcloud

Testing tool

- mdtest with a few changes
  - shared directory tests with multi-mount
  - mknod
- Test methodology
  - 1, 2, 4, 8, 16 clients
  - 1, 2, 4, 8, 16, 24, 32 threads on each client
    - Max to 512 client threads
      - Equal to max number of service threads on server
  - mknod, opencreate (0-stripe), opencreate (1-stripe), unlink
  - Shared directory
    - all client threads run under same target directory of different mount points, i.e: thread[0, 1, 2... 31] run under /mnt/lustre.[0, 1, 2... 31]/testdir
  - Directory per thread
    - Verify no performance drop for "directory per thread" case

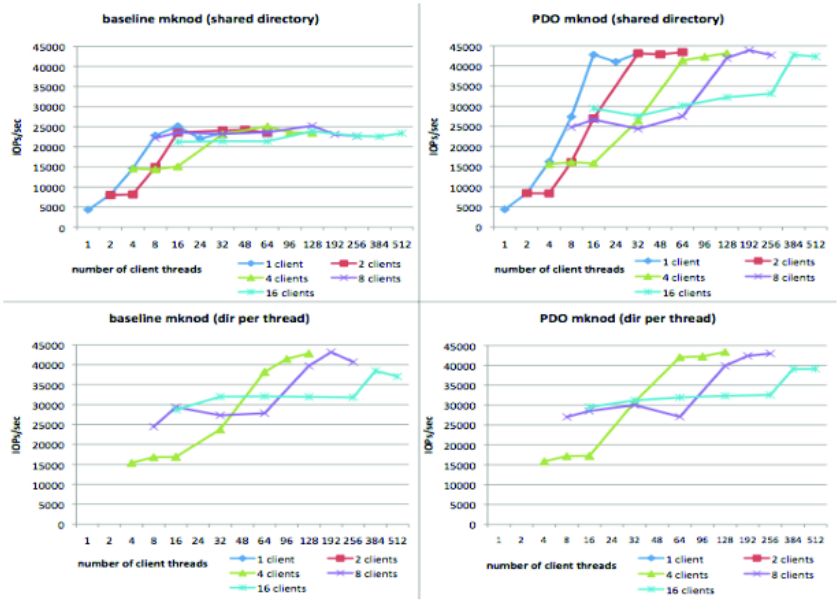
whamcloud

Mknod performance

- Total 1 million files
- Share directory
  - [1, 2, ..., 16] clients * [1, 2, ..., 32] threads
- Directory per thread
  - [4, 8, 16] clients * [1, 2, ..., 32] threads
- mknod is a single RPC
  - mknod performance can show PDO improvement with MDS stack

whamcloud

mknod performance (1 million files)

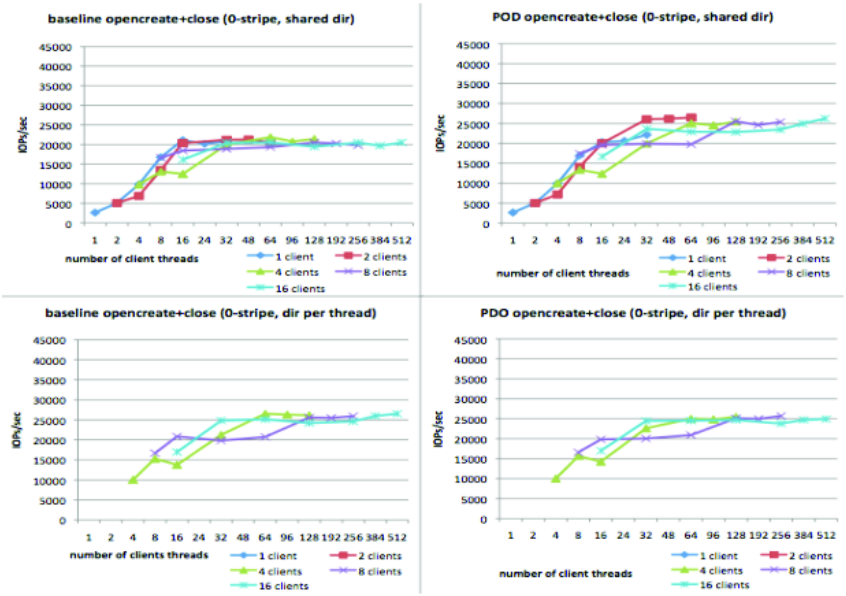


whamcloud

0-stripe Opencreate+close performance

- Total 1 million files
- Share directory
  - [1, 2, ..., 16] clients * [1, 2, ..., 32] threads
- Directory per thread
  - [4, 8, 16] clients * [1, 2, ..., 32] threads
- 2 RPCs for opencreate+close
  - PDO project can't help anything for "close"
  - Opencreate+close can't show full improvement on "create" because extra "close" RPC

0-stripe opencreate+close performance (1 million files)

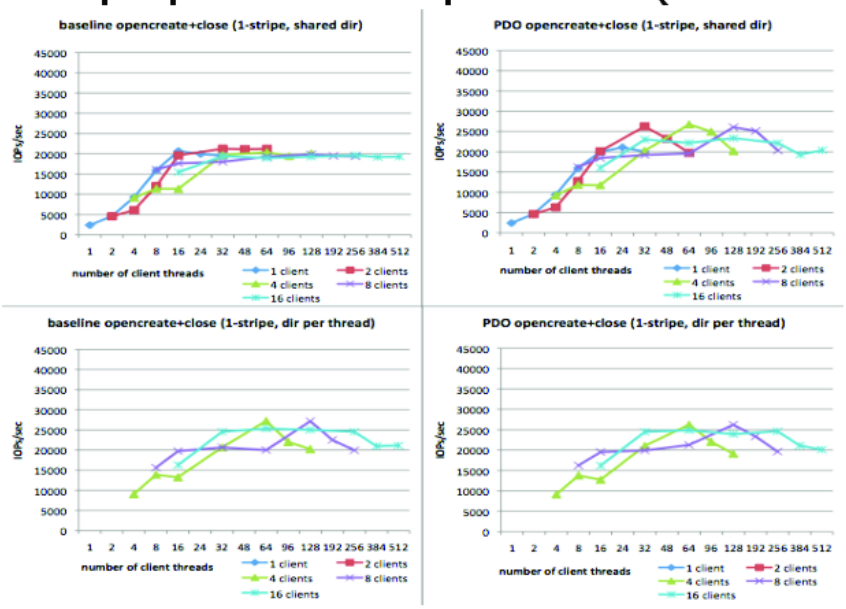


1-stripe Opencreate+close performance

- Total 1 million files
- Share directory
  - [1, 2, ..., 16] clients * [1, 2, ..., 32] threads
- Directory per thread
  - [4, 8, 16] clients * [1, 2, ..., 32] threads
- 2 RPCs for opencreate+close
  - PDO project can't help anything for "close"
  - More overhead

whamcloud

1-stripe opencreate+close performance (1 million files)



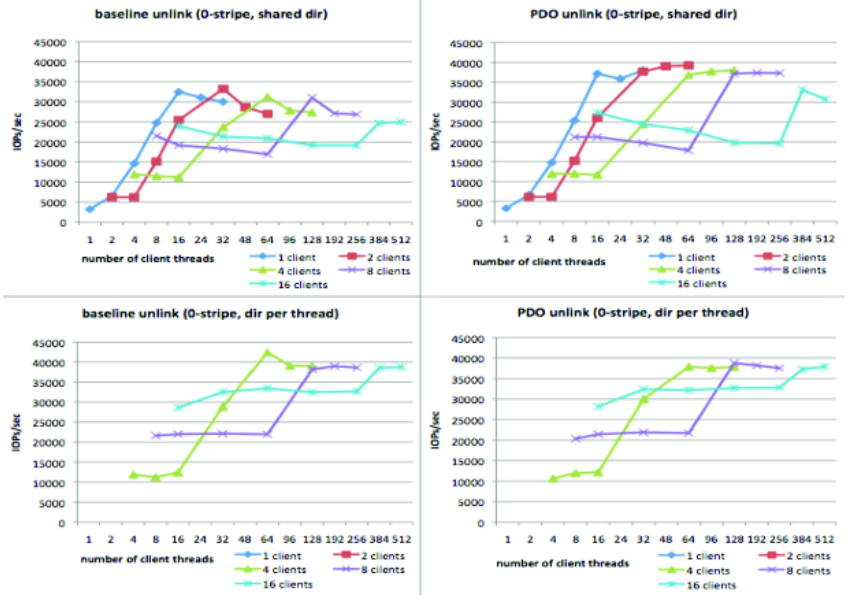
whamcloud

0-stripe file unlink performance

- Total 1 million files
- Share directory
  - [1, 2, ..., 16] clients * [1, 2, ..., 32] threads
- Directory per thread
  - [4, 8, 16] clients * [1, 2, ..., 32] threads

whamcloud

0-stripe file unlink performance (1 million files)



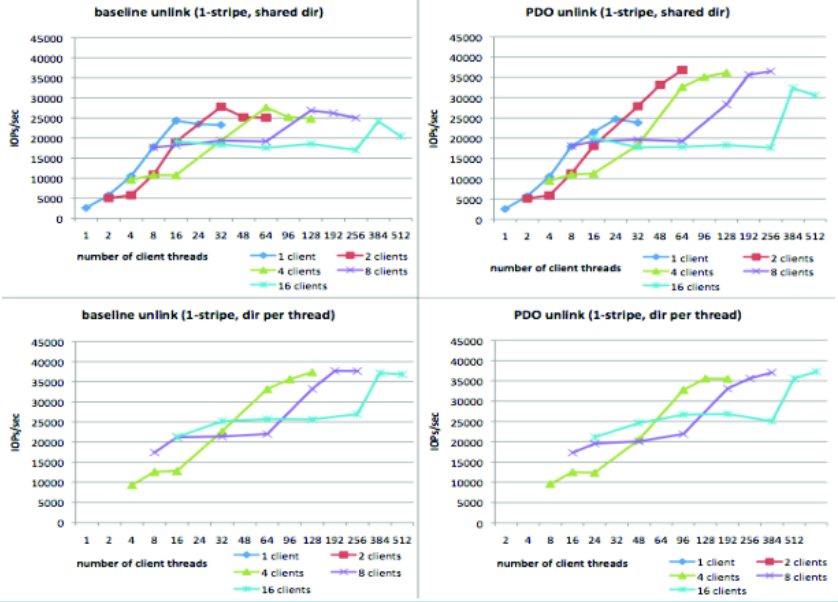
whamcloud

1-stripe file unlink performance

- Total 1 million files
- Share directory
  - [1, 2, ..., 16] clients * [1, 2, ..., 32] threads
- Directory per thread
  - [4, 8, 16] clients * [1, 2, ..., 32] threads

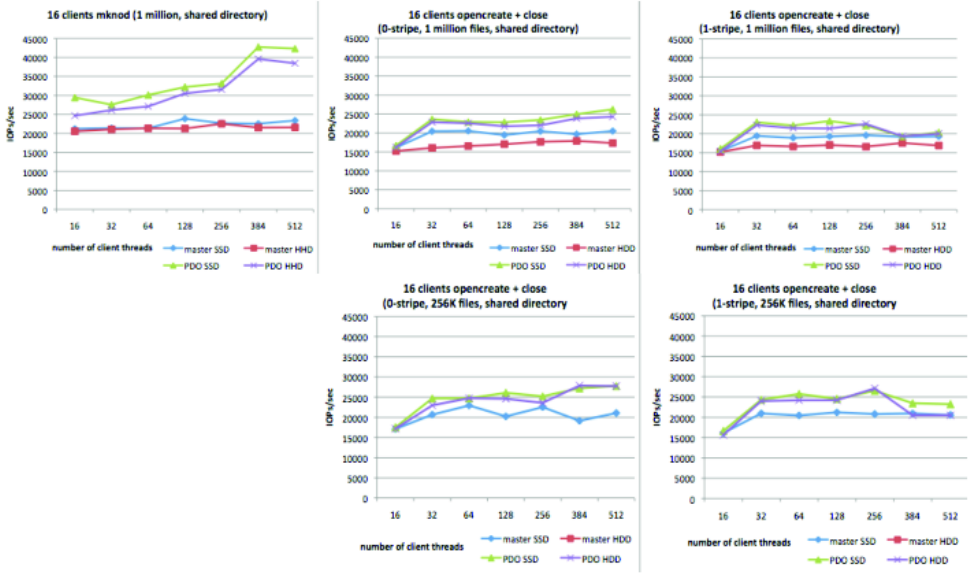
whamcloud

1-stripe file unlink performance (1 million files)



whamcloud

More comparisons: 16 clients creation performance



Summary

- PDO project improved performance of shared directory operations
  - mknod & unlink performance are significantly improved
    - Although there are some strange performance values in unlink tests, which need more survey
  - Opencreate+close is improved somehow
    - Can't show full improvement because of extra "close" RPC
    - Could have some performance issues in striped file, but it shouldn't be in scope of this project.
  - No performance drop for "directory per thread" case
  - Results could be better with SMP improvements