Metadata Test Harness Solution Architecture

1/20/12

Wang Di

Bryon Neitzel

Eric Barton

Revision History

Author	Ver	Date	Change
Wang Di <di.wang@whamcloud.com>	0.1	2011/08/08	Original version
Wang Di <di.wang@whamcloud.com>	0.2	2011/08/15	Update command-line interface, edits
Bryon Neitzel <bryon@whamcloud.com>	0.3	2011/08/10	Edits for clarity
Eric Barton <eeb@whamcloud.com>	0.4	2011/08/15	Clarify test cases, update interface, edits
Andreas Dilger <adilger@whamcloud.com>	0.5	2011/08/16	Changes to interface options
Wang Di <di.wang@whamcloud.com>	0.6	2011/08/17	Update command with test_xxx (suggested by Andreas)

Introduction

The investigation of Lustre MDS performance issues at scale currently requires the use of a large compute cluster like ORNL's Jaguar system. The cost and inconvenience of diverting computing resources normally engaged full time in production to filesystem development is unacceptable. This paper describes an alternative approach that exercises the MDS stack directly.

The Lustre team has previously developed tools such as *obdfilter-survey*, which tests OSTs directly on the OSS node without requiring connected clients. This tool uses the *echo_client* module, which attaches to the OSS stack to inject I/O operations at the obdfilter layer in the OSS.
We propose to implement analogous functionality for the MDS stack. This will generate load directly on the MDS as if driven by a large client cluster. For the purposes of this document the tool will be referred to as the *md-survey* tool. This will include modifications to the *echo_client* kernel module and to the *lctl* utility, which will in turn be driven by a performance survey script.

Solution Requirements

Create metadata test loads with no client nodes.

The *md-survey* tool will run directly on the MDS. It will generate a similar meta-data performance load on the MDS stack as is observed in full-scale client testing and drive the MDS to saturation. Three classes of operation are expected on files and directories.

- Open-create/mkdir
- Lookup/getattr/setattr
- Unlink/rmdir

These operations will be run by a variable number of concurrent threads and occur in a variable number of directories such that at one extreme, all threads execute in the same directory and at the other extreme, all threads execute in a private directory.
It will be possible to run *md-survey* with at least 4096 concurrent, which exceeds the number of MDS service threads used in production at large Lustre installations.

Exercise different layers of the metadata stack.

The md-survey tool should be able to inject tests at different levels of the MDS stack, so that performance issues at these different levels can be isolated, specifically:

- At the OSD API to measure OSD performance.
- At the MDD API to include MDD stack overhead.
- On top of the MDT layer to include the whole MDS stack.

While Whamcloud will anticipate this requirement in its implementation, the scope of this project will be limited to reproducing the performance issues previously seen in full scale testing. Whamcloud therefore proposes to start by targeting the MDD API since this will exercise full namespace operations and underlying filesystem code and hardware without the overhead and complexity of generating synthetic RPC requests. Tests that target the OSD API and the MDT layer will be implemented at a later time, unless required to replicate performance pathologies observed at full scale.

Solution Proposal

The *lctl* utility currently implements multi-threaded I/O tests using ioctl() to call into the *echo_client* kernel module. The *obdfilter-survey* shell script invokes *lctl* to iterate over different test types and parameters and collect results to build up a comprehensive picture of how the stack being exercised performs over a wide range of conditions.

Whamcloud proposes to implement *md-survey* using as much existing infrastructure as is appropriate. Specifically, the existing *lctl*/*echo_client* will be extended to include the metadata tests listed below, and a new *md-survey* script will be written to drive them.

All MDD-level metadata tests implemented in *lctl* will take the following parameters:

- `-d parent_basedir`
  `-D parent_dircount`
  These specify the number of directories in which to perform tests and their names. If the *parent_dircount* parameter is omitted, a single directory will be used. Otherwise the tests are spread over the given number of subdirectories designated by appending a numeric suffix to *parent_basedir*. Threads are assigned to directories in round-robin order.
- `-b child_base_id`
  This specifies the base name of the files or directories the tests will operation on. A numeric ID is appended to prevent name collisions on tests by concurrent threads.
- `-n count | -t time`
  These effectively specify how many test operations in total will be performed. Specifying *count* sets the number of operations to perform in each test thread. Alternatively, specifying *time* causes threads to run the given number of seconds.

The following *lctl* MDD-level metadata tests will be implemented:

- `lctl test_create -c stripe_count <generic-parameters>`
  This test creates regular files with the given mode and number of stripes. Specifying a *stripe_count* of 0 skips creating OST objects and associated overhead.
- `lctl test_destroy <generic-parameters>`
  This test deletes the files from the parent directories.
- `lctl test_mkdir <generic-parameters>`
  This test creates subdirectories in the parent directories instead of regular files.
- `lctl test_rmdir <generic-parameters>`
  This test removes the subdirectories from the parent directories.
- `lctl test_lookup <generic-parameters>`
  This test performs simple lookups that verify whether a file or directory exists.
- `lctl test_getattr <generic-parameters>`
  This test effectively does a stat(2) on the files or directories.
- `lctl test_setattr -m mode <generic-parameters>`
  This test sets attributes on the test files. The *-m mode* option sets the file permissions to the specified mode to simulate chmod(2).
- `lctl test_setxattr -x attr_size <generic-parameters>`
  The *-x attr_size* option stores an artificial attribute of *attr_size* bytes on each file. Specifying *attr_size* of 0 removes the artificial attribute from each file, if it exists.