Lustre Feature Test Plan for *Snapshot*

Revision: v1
Date: 04/07/2017

Table of Contents

Revision History

The following is a chronological history of changes made to this document.

Revision	Date	Reason for change	Author
v.1	04/07/2017	Initial Version	Saurabh Tandan

Introduction

Snapshots provide fast recovery of files from a previous checkpoint (without recourse to offline backup). Snapshots are cheap online backups, provided the hardware itself is not compromised. Recovery of lost files from a snapshot is usually considerably faster than from any offline backup or remote replica. It is noted that snapshots do not improve storage reliability and are just as exposed to hardware failure as any other storage volume.

Snapshot addresses a need to be able to take a checkpoint of the file system, and has two historic purposes: prepare a file system for a backup or fast recovery of files from a previous state without recourse to an offline backup. The latter option is increasingly used in environments where the cost associated with any downtime is significant – consider the time required to restore a dataset from a tape library. In many cases, restore from tape will exceed the SLA for operations.

Stabilizing the file system for a backup is probably less relevant when the file system size reaches into petabytes. LTO drives for example, can only record at a maximum rate of 576-900GB/hour. As file system capacities increase, the ability to take a backup, and more importantly restore from a backup within the conditions of an SLA, diminish.

Let us not underestimate the utility of snapshots when planning maintenance. Taking a snapshot immediately prior to a system upgrade is a sensible precaution and making that mechanism accessible and reliable adds value to any system maintenance workflow.

Lustre Snapshot contains four mains components:

1. Kernel level write (or say 'modification') barrier on all MDTs .

   Before creating the snapshot, we need to setup write barrier on all MDTs to guarantee that when the barrier is setup successfully, then:
   1.1) All the pending destroy/chown requests have been flushed from the MDTs to the OSTs.
   1.2) No new OST-object pre-create requests from the MDTs to the OSTs.
   1.3) All on-going modifications on MDTs have been handled.
   1.4) New modification requests from clients to MDTs will be blocked.
   NOTE: The barrier does not guarantee the OST-side data consistence because we cannot properly control client-side application's I/O (cache) behavior. That is a known issue.

2. Fork/erase Lustre configuration

   It is required that the snapshot and its original filesystem can be mounted with shared MGS, so the snapshot must use different FSNAME from its original filesystem. The most direct solution is to clone the original filesystem configuration logs and replace the FSNAME instances in the configuration logs with new FSNAME when create the snapshot. Similarly, when removing the snapshot, its configuration on the MGS needs to be destroyed.

3. Mount Lustre device as read only mode

   Lustre snapshot is read only. Once it has been created, its content cannot be modified any more (**NOTE:** its attribute can be modified via ***lsnasphot***). There are many self-sponsored and recovery-triggered modifications during the mount. All these need to be bypassed or disabled when mount Lustre snapshot.

4. User space interfaces/commands

   The Lustre snapshot will utilise ZFS backend snapshot functionalities to build Lustre snapshot tools set, called ***lsnapshot***. They will be implemented as a series of lctl commands.

This feature is being tracked at LU-8900.

For more detailed information about Lustre snapshot, please reference the document: Filesystem Snapshots.

Use cases

The typical use cases of Lustre snapshot are as following:

Case 1: create snapshot for async backup

Backup Lustre filesystem online is a complex task. As the filesystem size increases, it will take the backup tool more time to complete. It is quite common that some other users are modifying the system during the backup. So it is ambiguous to define the backup time-point and the content that has been backup. With Lustre snapshot introduced, things become clear and easy. The snapshot defines a check-point, the backup tool can backup the system against the snapshot instead of the original filesystem. Since the snapshot is readonly. Nobody will modify it even if someone modified the original system after such check-point. So the backup tool can deliberately backup the system under asynchronous mode.

Case 2: create snapshot for fast recovery

Currently, there is no efficient Lustre data recovery mechanism if you remove such file explicitly. That is different from the user experience of finding back the file from "Trash" under desktop environment. In further, even if there is "Trash" mechanism for Lustre, but if the file data is modified by wrong, it is still trouble to recover the data. Recovering the file for the backup is the solution. Traditionally, it is inefficient to find out the specified version of file from the backup that is possibly stored on slow device. But in case of Lustre snapshot it is not. It shares the same device as the original system, and can run in separated namespace with the original filesystem online. Then it is quite easy to copy the specified version file from the snapshot to recover the file in the original filesystem. Typically, the sysadmin can create snapshot periodically, such as per hour (or per day, and so on).

Case 3: create snapshot for before upgrade

Upgrading Lustre is unavoidable, whether it is for main system upgrading or for bug fixing. Regardless of this, it is sensible and good practice to backup the system before the upgrading, especially when the upgrading involves on-disk data layout adjustment as some changes cannot be rolled back, such as enabling "dirdata". If the change made by the upgrade causes some trouble, such as server inaccessible or interoperability issues, then downgrade the system and start the system from backup. Without Lustre snapshot, both backup and restore will be time-consuming tasks, depending on the system size. But, with Lustre snapshot support, there is no need to backup the system to other storage, we trust that the upgrading will not damage the storage hardware, hence, backing up the original system before the upgrading may take only seconds or a few minutes at most. And once you need to restore the system, just copy the data from the snapshot after downgrade.

Case 4: recover the file from backup

Corresponding to the case 2, when it is required to recover the file from backup, then mount up both the original filesystem and the snapshot (the MGS is shared), and then find out the required file(s) from the snapshot, and copy it to the original filesystem to replace the current one(s).

Case 5: restore the system from backup

Corresponding to the case 3, when required to restore the whole Lustre system from the backup there can be following cases - either the original system cannot be mounted, or there are too many things to be recovered and recovering them one by one is more time-consuming than recovering the whole system together. Under such cases, format a new Lustre system is necessary (not destroy the snapshot), then mount the new filesystem, and then copy all the data from the snapshot to the new filesystem.

Feature Overview

Installation

lsnapshot is community release special, the target is Community 2.10. No special requirement for installation.

Configuration

The system config information, such as each target's hostname, pool name, local filesystem name, role, index, and so on, will be written in the snapshot config file " /etc/ldev.conf", if it does not exist, then uses the file "/etc/lsnapshot/${fsname}.conf". The format for "/etc/ldev.conf" is as following:

```
<host> foreign/- <label> <device> [journal-path]/- [raidtab]

 The format of <label> is:
fsname-<role><index> or <role><index>

 The format of <device> is:
[md|zfs:][pool_dir/]<pool>/<filesystem>

Snapshot only uses the fields <host>, <label> and <device>.

For example:

host-mdt1 - myfs-MDT0000 zfs:/tmp/myfs-mdt1/mdt1
```

The format for "/etc/lsnapshot/${fsname}.conf" is as following:

host pool_dir pool local_filesystem role(,s) index

The administrator can edit the config file manually or generate it automatically via some scripts.

How to enable the feature

There is no specific command required to enable this feature. It is always on.

How to disable the feature

N/A

How to monitor the feature

This feature can be monitored with the help of **`lctl`** commands (see below). The logs for this feature are collected at **`/var/log/lsnapshot.log`** , which can be used to monitor the functioning of this feature.

Test cases

Unit testing

N/A

Functional Testing

The functional tests have been done by the scripts under lustre/tests/, including:

Functional Testing for lustre writer barrier

lustre/tests/sanity.sh test_801{a,b,c} (see the patch https://review.whamcloud.com/#/c/24265/)

- Barrier stat machine: freeze barrier, thaw barrier, barrier timeout, barrier failure, and so on.
- Try to modify the system with barrier frozen.
- Rescan the system to update barrier bitmap under the case of some MDT(s) unavailable

Functional Testing for read only device

lustre/tests/sanity.sh test_802 (see the patch https://review.whamcloud.com/#/c/24267/)

- Simulate readonly device, and try to modify the system under read only mode.

Functional Testing for *lsnapshot*

lustre/tests/sanity-lsnapshot.sh (see the patch https://review.whamcloud.com/#/c/24269/),

Test Description	**Test-ID in sanity-lsnapshot.sh**
Create snapshot with default option	test_0
Create snapshot with barrier off	test_0
Create snapshot with comment	test_0
Try to create the snapshot that exists already	test_0
List the snapshot	test_0
List the snapshot with detail	test_0
List all snapshots of the filesystem	test_0
Mount the snapshot with original MGS online	test_1a
Mount the snapshot with original MGS offline	test_1b
Umount the snapshot	test_1a, test_1b, test_3a

Destroy the snapshot with default options	test_0
Try to destroy the snapshot that is mounted	test_0
Destroy the snapshot by force	test_2
Modify the snapshot's name	test_3a
Modify the snapshot's comment	test_3a
Modify the snapshot's name and comment together	test_3a
Try to modify the snapshot's name that is mounted	test_3a
Modify the snapshot's comment that is mounted	test_3a
Modify the snapshot with original MGS offline	test_3b

Regression Testing

All the existing sanity test cases under lustre/tests/ should pass except some known bugs.

Scalability Testing

We need to test how long it will take to freeze barrier on all MDTs. It is expected that the time will increase as the MDTs count increases. And it is affected by system load also. Currently, we still do not know how long is the acceptable time. We need to measure that first. If it is too long, such as several minutes, we may have to consider re-implement the writer barrier.

We need to measure the time of freezing barrier on the system with 1/2/4/8/16 MDTs cases, with and without system metadata load (mdtest). This should also be tested under following two scenarios:

1. 1 MDT per MDS with multiple MDS in the system
2. Multiple MDT per MDS with one MDS in the system

Feature Compatibility

DNE-II

The Lustre snapshot is compatible with DNE-II feature and testing for this feature is covered above in Scalability testing when testing:

1. 1 MDT per MDS with multiple MDS in the system
2. Multiple MDT per MDS with one MDS in the system

Subdirectory Mount

The Lustre Snapshot is compatible with Subdirectory Mount feature and it can be tested as follows:

1. Build a simple Lustre File System with 1 MDT, 1 OST and 1 Client
2. Have a subdirectory mounted in a Lustre File System. Please refer here for Subdirectory Mount Feature.
3. Create a Snapshot.
4. List a snapshot.
5. Destroy a Snapshot.

Interoperability Testing

Lustre snapshot works on server-side only. The interoperability issues are among MDTs/OSTs. If we do not want to support server side interoperability (meaning not upgrading MDTs/OSTs), then no need to consider this.

Performance Testing

ZFS filesystem modification is based on copy-on-write (COW), which makes the modification with snapshot created to be easy. This is because ZFS will write the modification to new block(s) regardless of whether there is a snapshot or not. So it is expected that modifying the zfs filesystem with snapshot created will cause very little overhead and can be ignored. We can verify that via IOR/mdtest by comparing the cases of with/without snapshot created.

Failure and recovery testing

All existing failure and recovery tests will be run -
https://wiki.hpdd.intel.com/display/ENG/Regression+Test+Suites+and+Failover+Test+Suites

User Interface Components

The user interfaces for Lustre snapshot are implemented as part of lctl tools set, please check the section "lctl commands added or changed" for detail.

Procfs/Sysfs entries added or changed:

N/A

Other system settings added or changed:

- new configuration file /etc/ldev.conf for ***lsnapshot***
- new log file /etc/log/lsnapshot.log for ***lsnapshot***

lfs commands added or changed:

N/A

lctl commands added or changed:

lctl commands for writer barrier

- Freeze barrier:
  lctl barrier_freeze <fsname> [timeout (in second)]
  NOTE: the default timeout value is 30 (seconds).
- Thaw barrier:
  lctl barrier_thaw <fsname>
- Query barrier:
  lctl barrier_stat <fsname>
- Rescan barrier bitmap:
  lctl barrier_rescan <fsname> [timeout (in second)]
  NOTE: the default timeout value is 30 (seconds).

lctl commands for MGS config

- To fork a configuration log, run the following lctl command on the MGS:
  Usage: lctl fork_lcfg <fsname> <newname>
- To erase a configuration log, run the following lctl command on the MGS:
  Usage: lctl erase_lcfg <fsname>

lctl commands for snapshot

- Create the snapshot
  lctl snapshot_create [-b | --barrier [on | off]] [-c | --comment comment] <-F | --fsname fsname> [-h | --help] <-n | --name ssname> [-r | --rsh remote_shell][-t | --timeout timeout]
  Options:
  -b: set write barrier before creating snapshot, the default value is 'on'.
  -c: describe what the snapshot is for, and so on.
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.
  -r: the remote shell used for communication with remote target, the default value is 'ssh'.
  -t: the life cycle (seconds) for write barrier, the default value is 30 seconds.
- Destroy the snapshot
  lctl snapshot_destroy [-f | --force] <-F | --fsname fsname> [-h | --help] <-n | --name ssname> [-r | --rsh remote_shell]
  Options:
  -f: destroy the snapshot by force.
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.
  -r: the remote shell used for communication with remote target, the default value is 'ssh'.
- Modify the snapshot
  lctl snapshot_modify [-c | --comment comment] <-F | --fsname fsname> [-h | --help] <-n | --name ssname> [-N | --new new_ssname] [-r | --rsh remote_shell]
  Options:
  -c: update the snapshot's comment.
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.

-N: rename the snapshot's name as the new_ssname.
-r: the remote shell used for communication with remote target, the default value is 'ssh'.

- Query the snapshot(s)
  lctl snapshot_list [-d | --detail] <-F | --fsname fsname> [-h | --help] [-n | --name ssname] [-r | --rsh remote_shell]
  Options:
  -d: list every piece for the specified snapshot.
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.
  -r: the remote shell used for communication with remote target, the default value is 'ssh'.
- Mount the snapshot
  lctl snapshot_mount <-F | --fsname fsname> [-h | --help] <-n | --name ssname> [-r | --rsh remote_shell]
  Options:
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.
  -r: the remote shell used for communication with remote target, the default value is 'ssh'.
- Umount the snapshot
  lctl snapshot_umount <-F | --fsname fsname> [-h | --help] <-n | --name ssname> [-r | --rsh remote_shell]
  Options:
  -F: the filesystem name.
  -h: for help information.
  -n: the snapshot's name.
  -r: the remote shell used for communication with remote target, the default value is 'ssh'.

New Scripts added:

- lustre/test/sanity-lsnapshot.sh

Man Pages entries added or changed:

See the man page patch https://review.whamcloud.com/#/c/24270/

Updates to test infrastructure:

N/A