Lustre disaster recovery with robinhood 2.5

2013, November 27th

Table of contents

Disclaimer

Robinhood disaster recovery tools for Lustre are shared with the community with the hope that it will be helpful to recover from catastrophic situations.

These tools are distributed WITHOUT ANY WARRANTY and their use is AT YOUR OWN RISK.

It is highly recommended that you validate the disaster recovery process described in this document with your Lustre support before using it on a production system.

Theses tools have only been tested with a single Lustre version and operating system (Lustre 2.1.0 on CentOS 6.1). They may require adaptations to work in other configurations.

Introduction

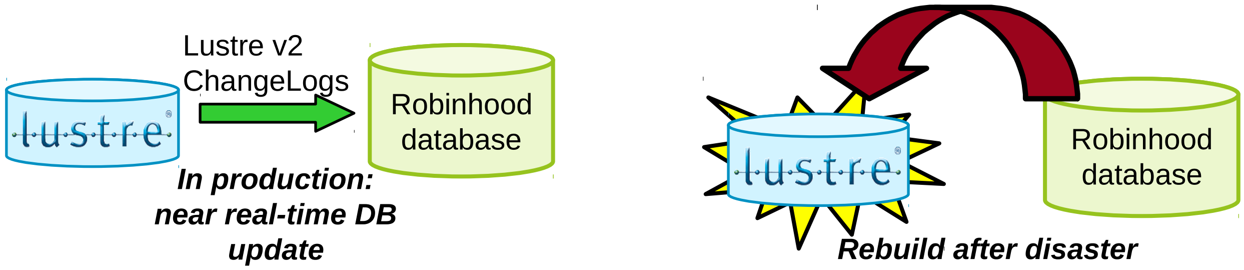
Thanks to the various modes of Robinhood PolicyEngine, Lustre filesystems can be integrated in different ways in a storage infrastructure:

- as a standalone “scratch” filesystem
- as a filesystem whose data is archived to a backup storage
- as a front-end to a hierarchical storage system (HSM)

Each of these configurations uses a different Robinhood flavor:

Filesystem configuration	Robinhood flavor
Standalone Lustre	robinhood-tmpfs
Lustre+Backup	robinhood-backup
Lustre/HSM	robinhood-lhsm

By reading Lustre v2 changelogs, Robinhood maintains a replicate of filesystem metadata updated near real time. This information can be useful to rebuild a filesystem after a disaster (hardware failure, data corruption...).



This document describes the possible solutions that can be considered to recover from various disaster scenarios in each of the configurations listed above.

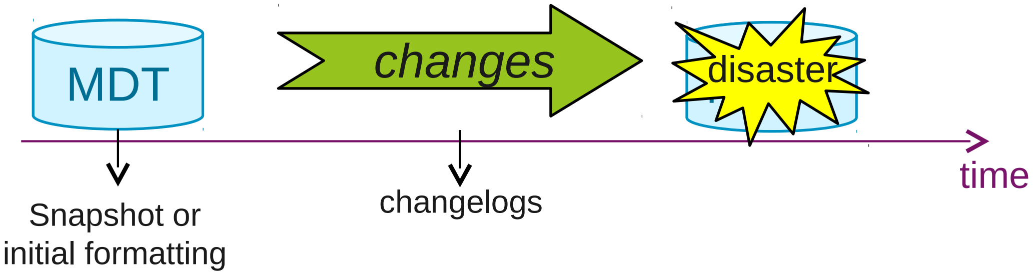
MDS disaster recovery

Overview

Recovery tools make it possible to rebuild a Lustre MDT after its content has been lost or corrupted. In this case, OSTs still contain valid data objects, but they are no longer accessible. The goal of this procedure is to restore MDT objects with up-to-date attributes and paths at the time of the disaster, and restore the relationships between existing OST objects and MDT objects, so the valid data on OSTs can be preserved and accessed again.

This procedure allows rebuilding a MDT from scratch, or starting from a snapshot (eg. created with LVM) which speeds up the recovery operation.

Requirement: before the disaster occurs, robinhood must be configured and must be running on the filesystem. If it configured for reading Lustre Changelogs, it will then be able to restore the MDT state at the time of the last Changelog record it read. If is only configured for scanning the filesystem, it can restore the filesystem state at the time of the last scan.



How it works

Robinhood 2.5 replicates most Lustre metadata in its database, for all object types (files, directories, symlinks...):

- Posix attributes: type, owner, group, atime, mtime, size, path (parent+name), access rights, symlink definitions.
- Striping information: stripe count, stripe size, OST pool name, ordered list of stripe objects (ost_idx, ost_gen, obj_id, obj_seq).

After a MDT is lost:

- recovery tools prevent from cleaning orphaned OST objects, as robinhood is aware of the existing objects on OSTs.
- with the new command **rbh-diff**, robinhood detects and lists differences between the current MDT state (empty, or outdated snapshot) and the information it has in its database.
- It can then apply on the MDT:
  - recent changes in object attributes and paths
  - remove recently deleted files
  - restore newly created objects in the namespace, and relink then with existing data on OSTs.

Recovery procedure

1. Restore the MDT device in a consistent state:
   - from scratch: format a new MDT
   - from a snapshot (if managed with LVM):
     e.g. `dd if=/backup/snap-mdt0-20130528 of=/dev/vgmdt-mdt0`
2. Before starting Lustre, we must prevent it from cleaning orphan objects on OSTs. This is done by generating replacing the *lov_objid* file on the MDT:
   - Generate a new *lov_objid* using command **gen_lov_objid** (provided by robinhood-recov-tools):
     ```
     gen_lov_objid -o /tmp/lov_objid.new
     ```
     (validate the contents of the generated file with your Lustre support).
   - Mount the MDT in ldiskfs mode:
     ```
     mount –t ldiskfs /dev/vgmdt-mdt0 /mnt/mdt
     ```
   - Save the previous lov_objid, and replace it with the new one:
     ```
     cp /mnt/mdt/lov_objid /mnt/mdt/lov_objid.bkp
     cp /tmp/lov_objid.new /mnt/mdt/lov_objid
     ```
   - Unmount the MDT

3. Applying metadata changes to the filesystem:
   - Start Lustre and mount a client on the robinhood host
   - Use **rbh-diff** to apply recent metadata changed, object deletions and creations. The command generates some data in an output directory

this will be used for next steps.

```
mkdir /tmp/recov_info
rbh-diff –-apply=fs –o /tmp/recov_info
```

- The command displays all the detected metadata differences, and restore the state as in robinhood DB.
- Metadata differences are represented as a 'diff -u' output, with the list of modified attributes per object.
  Example:

  ```
  -[0x200000bd0:0x11:0x0] mode=0750
  +[0x200000bd0:0x11:0x0] mode=0700
  - [0x200000bd0:0x1a:0x0] path='/mnt/lustre/file'
  +[0x200000bd0:0x1a:0x0] path='/mnt/lustre/fname'
  ```

- Object creations are displayed as lines starting with '++', with the list of entry attributes.
  e.g.

  ```
  ++[0x200000bd0:0xf:0x0] path='/mnt/lustre/dir.new',parent=[0x61ab:0x2dd8dd16:0x0],
  type=dir,nlink=2,mode=0755,owner=root,group=root,size=4096,blocks=8,depth=0,dircount=0,
  access=1383128424,modif=1383128424,creation=1383128424
  ```

- Object deletion are displayed as lines starting with '--'.
- rbh-diff generates 2 files in the output directory: *lovea* and *fid_remap*. They must be carefully preserved for the next steps.

4. Restoring MDT<->OST objects relationships:
   - Unmount and stop Lustre
   - Mount the MDT device in ldiskfs:

     ```
     mount –t ldiskfs /dev/vgmdt-mdt0 /mnt/mdt
     ```

   - Restore MDT -> OST objects relationship using command **set_lovea** (provided by robinhood-recov-tools) with the *lovea* file generated at step 3:

     ```
     set_lovea /mnt/mdt /tmp/recov_info/lovea
     ```

   - Restore OST->MDT relationship (not required for going back to production, but required to avoid *lfsck* to go crazy).
     This operation can be performed on all OSTs in parallel.
     - Mount a given OST in ldiskfs:

       ```
       mount –t ldiskfs /dev/ostXXX /mnt/ostXXX
       ```

     - Use **ost_fids_remap** command to fix OST->MDT relationships, with the *fid_remap* file generated at step 3:

       ```
       ost_fids_remap idx /mnt/ostidx /tmp/recov_info/fid_remap
       ```

When restarting Lustre, all objects are restored and their data is accessible.

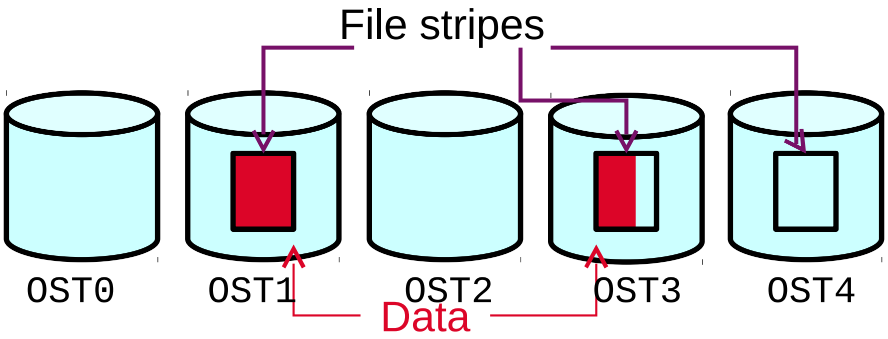
OST disaster recovery

After loosing the content of an OST, the recovery solution greatly depends on your robinhood setup:

- if no backup or external copy (HSM) of data is available, the data on the lost OST is definitely lost.
  In this case, robinhood can still help by identifying the impacted files.
- with backup and HSM configurations, robinhood can:
  - identify the impacted files
  - restore the data from the archive (possibly an old version)
  - summarize what could be restored at the latest state, at a previous state, and what couldn't be restored at all.

Identifying the impacted files

Even if a file is striped on a given OST, it doesn't mean the file has data on it (depending on file size, stripe count, stripe width, stripe order, ...). This is illustrated in the figure below:



Since robinhood 2.5, “`rbh-report --dump-ost`” now indicates whether files really have data on a given OST (or set of OSTs).

For example, if we loose OSTs 2, 5, 6, 7 and 8, we can use the following command and look at the last column:

```
> rbh-report --dump-ost 2,5-8

type,     size,                path, strp_cnt, strp_size,  stripes,                   data_on_ost[2,5-8]
file,  8.00 MB,   /fs/dir.1/file.8,        2,   1.00 MB,  ost#2: 797094, ost#0: 796997,          yes
file, 29.00 MB,   /fs/dir.1/file.29,       2,   1.00 MB,  ost#2: 797104, ost#0: 797007,          yes
file,  1.00 MB,   /fs/dir.4/file.1,        2,   1.00 MB,  ost#3: 797154, ost#2: 797090,           no
file, 27.00 MB,   /fs/dir.1/file.27,       2,   1.00 MB,  ost#3: 797167, ost#2: 797103,          yes
file, 14.00 MB,   /fs/dir.5/file.14,       2,   1.00 MB,  ost#3: 797161, ost#2: 797097,          yes
file, 13.00 MB,   /fs/dir.7/file.13,       2,   1.00 MB,  ost#2: 797096, ost#0: 796999,          yes
file, 24.00 KB,   /fs/dir.1/file.24,       2,   1.00 MB,  ost#1: 797102, ost#2: 797005,           no
```

➔ If a file is striped on a lost OST, but doesn't look impacted in the previous report, it can be read and it just needs to be re-striped to sane OSTs (using lfs migrate, for example).

- ➔ If a file is impacted, the solution depends on the robinhood mode you use:
  - with *tmpfs*, the only solution is to remove the files and send a mail to the file owner...
  - for backup and HSM modes, there is more hope: see the next section.

Recovering data from backup or HSM

In a backup or HSM configuration, “rbh-report --dump-ost” displays an extra indication about file status: *new*, *synchro*, *modified* or *released*.

```
> rbh-shook-report --dump-ost 2
status,       size,              path, strp_cnt, stripe_size,                      stripes, data_on_ost2
new,        8.00 MB,  /fs/dir.1/file.8,       2,     1.00 MB,  ost#2: 797094, ost#0: 796997,         yes
synchro,   29.00 MB,  /fs/dir.1/file.29,      2,     1.00 MB,  ost#2: 797104, ost#0: 797007,         yes
released,   1.00 MB,  /fs/dir.4/file.1,       2,     1.00 MB,  ost#3: 797154, ost#2: 797090,          no
released,  27.00 MB,  /fs/dir.1/file.27,      2,     1.00 MB,  ost#3: 797167, ost#2: 797103,          no
synchro,   14.00 MB,  /fs/dir.5/file.14,      2,     1.00 MB,  ost#3: 797161, ost#2: 797097,         yes
new,       13.00 MB,  /fs/dir.7/file.13,      2,     1.00 MB,  ost#2: 797096, ost#0: 796999,         yes
modified,  24.00 KB,  /fs/dir.1/file.24,      2,     1.00 MB,  ost#1: 797102, ost#2: 797005,          no
```

- **New** files: newly created files that have not yet been copied to the backend storage.
  They can’t be recovered if they had data in the lost OST.
- **Synchro** files: their contents have been copied to the backend and have not been modified in Lustre (the data in the backend storage is the same as the data in Lustre).
  Their contents can be recovered from the backend.
- **Modified** files: their contents have been copied to the backend and have been modified in Lustre (the data in the backend storage in an old version).
  A previous version of file contents can be recovered from the backend.
- **Released** files (shook and HSM modes only): their contents are only in the backend storage (no longer in Lustre). They have no data in Lustre: they just need to be restriped to safe OSTs. This case should not appear with Lustre-HSM, as released files have no stripe objects.

Recovery strategies

There are several possible recovery strategies for each mode. These strategies take a different time to put back the system to production and may result in a different load on the backend storage.

Strategies for **backup** mode:

1) Favor copy from the backend: in this case, we remove *synchro* objects striped on lost OST, even if they had no data on it, and then copy them back from the backend.

2) Favor copy from Lustre to Lustre: if a file has valid data in Lustre and is striped on the lost OST, we restripe it by copying data from Lustre to Lustre.

Most of the time, the 2nd solution will be faster, assuming that the Lustre throughput is higher than the backup storage. Thus, we will focus on describing the recovery mechanism for the 2nd strategy.

Strategies for **HSM** mode:

1) Favor releasing files: this makes the recovery operation a metadata-only operation, which speeds-up the return to production. The side effect is a higher load on the backend storage to recover file data on next access.
2) Favor copy from Lustre to Lustre: favor retrieving data in Lustre when possible. This requires copying data to restripe files, but avoids bringing data back from the backend on next access.

The time to go back to production is a critical point in a disaster recovery process. Thus, we will focus on the 1st strategy.

Recovery procedure for *backup* mode

1) Disable access from clients for the lost OST, to avoid creating new files on them.
2) List impacted object using “rbh-backup-report --dump-ost <*ost_idx*>“
3) Restripe all files striped on the lost OST, but with no data on it (right column in the previous report = ‘no’).
    a. Make sure Robinhood Changelog reader is running, to be aware of the restripe operation.
    b. Copy each file to a new file, and remove the file striped on the lost OST.
4) All remaining files have lost data. They must be retrieved from the backend (when possible).
    a. Shutdown Robinhood Changelog reader (Robinhood must keep info about those files when you will remove them).
    b. Using “rbh-backup-report --dump-ost <*ost_idx*>“, list the remaining objects from the lost OST and remove them from Lustre (rm).
    c. Perform a recovery for this OST, using ‘**rbh-backup-recov**’ (see ANNEX).

The OST disaster recovery in now completed. The failed OST no longer has files on it, and all the possible data has been recovered.

Recovery procedure for *HSM* mode

1) Disable access from clients for the lost OST, to avoid creating new files on them.
2) Release all possible objects from the dead OST:
```
rbh-lhsm --purge-ost=idx,0 –ignore-policies
```

    Remaining objects on the OST only have status ’*new*’, ’*modified*’ or ‘*released*‘:

```
> rbh-lhsm-report –dump-ost
```

```
status,      size,                path, strp_cnt, strp_size,                      stripes, data_on_ost2
new,       8.00 MB,   /fs/dir.1/file.8,       2,     1.00 MB,  ost#2: 797094, ost#0: 796997,        yes
released,  1.00 MB,   /fs/dir.4/file.1,       2,     1.00 MB,  ost#3: 797154, ost#2: 797090,         no
released, 27.00 MB,   /fs/dir.1/file.27,      2,     1.00 MB,  ost#3: 797167, ost#2: 797103,         no
```

```
modified, 13.00 MB,   /fs/dir.7/file.13,      2,     1.00 MB,  ost#2: 797096, ost#0: 796999,        yes
modified, 24.00 KB,   /fs/dir.1/file.24,      2,     1.00 MB,  ost#1: 797102, ost#2: 797005,         no
```

3) Restripe all the files displayed above with no data on the lost OST (right column = ‘no’), using *lfs migrate*.
4) Remaining entries at this step are ‘new’ or ‘modified’ entries with data on the lost OST. They must be retrieved from the backend (when possible):
   a. Shutdown Robinhood Changelog reader (Robinhood must keep info about those files when you will remove them).
   b. Using “rbh-lhsm-report --dump-ost <*ost_idx*>“, list the remaining objects from the lost OST and remove them from Lustre (rm).
   c. Perform a recovery for this OST, using ‘**rbh-lhsm-recov’** (see ANNEX).

The OST disaster recovery in now completed. The failed OST no longer has files on it, and all the possible data has been recovered.

ANNEX: using rbh-recov

Notice: robinhood changelog reader must be stopped during this operation.

1) **rbh-recov**[1] **--start** initializes the recovery procedure. This indicates how many entries can be fully recovered, how many files can be recovered with old data, and how many files can’t be recovered because they have no archived data.
   This command can be restricted to a single OST by specifying a ‘--ost’ option.

   > **rbh-recov --start** (recover the entire filesystem)
   or
   > rbh-recov --start --ost=<*ost_idx*> (recover an OST or set of OSTs)

```
Recovery successfully initialized.

It should result in the following state:
   - full recovery: 15522 files (60.61 TB), 0 non-files
   - old version:             193 entries (13.62 GB)
   - not recoverable:         964  entries (1.43 TB)
   - other/errors:            0/0 (0)
```

2) **rbh-recov --resume** then run the recovery operation. To be faster, the processing can be parallelized on several Lustre clients. Each of them is in charge of a part of the namespace:

```
client1> rbh-recov --resume -D /fs/cont001
client2> rbh-recov --resume -D /fs/cont002
client3> rbh-recov --resume -D /fs/cont003
…
```

[1] In this section rbh-recov stands for rbh-backup-recov for backup mode, and rbh-lhsm-recov for Lustre/HSM.

3) You can use **rbh-recov -status** to show the current progress of the recovery operation:

```
> rbh-recov --status
```

```
Current recovery status:
   - successfully recovered: 3582 files (9.82 TB), 0 non-files
   - old version:              54 entries (5.28 GB)
   - not recoverable:          252 entries (221.68 GB)
   - errors:                   2 entries (3.21 KB)
   - still to be recovered:    12789 entries (52.01 TB)
```

To get the detailed list of files, use "rbh-recov --list":

```
> rbh-recov  --list all
```

```
   type     state           path                           size
   file     todo            /mnt/lustre/dir.22/file        42.80 GB
   file     done            /mnt/lustre/dir.24.file.x      12.25 GB
   dir      done_non_file   /mnt/lustre/dir.25             4.00 KB
   file     done_empty      /mnt/lustre/dir.234/file.y     0
```

4) If recovery operations fail for any transient reason, they can be retried using the "--retry" option:

```
> rbh-recov --resume --retry -D /fs/cont003
```

5) End of recovery:

   a. When all recoveries operations are done, you can get the detailed list of erroneous entries using "rbh-recov --list"**:**

```
> rbh-recov --list failed
```

```
type     state           path                                size
file     failed          /mnt/lustre/dir.1/file.1.rnm        52.10 GB
file     failed          /mnt/lustre/dir.2/file.2.rnm        2.28 GB
file     failed          /mnt/lustre/dir.2/file.2            1.24 KB
file     failed          /mnt/lustre/dir.2/link_file         0
```

   b. Then, complete the recovery operation
   (/!\ This results in cleaning the content of the RECOVERY table, so the previous entry list is lost)

```
> rbh-recov --complete
```

```
Recovery successfully completed:
   - successfully recovered: 15522 files (60.61 TB), 0 non-files
   - old version:              193 entries (13.62 GB)
   - not recoverable:          962  entries (1.43 TB)
   - errors:                   2 entries (3.21 KB)
```