TSM Copytool for Lustre HSM

Thomas Stibor

\texttt{t.stibor@gsi.de}

High Performance Computing
GSI Helmholtz Centre for Heavy Ion Research
Darmstadt, Germany

Tuesday 20\textsuperscript{th} September, 2016

LAD 2016 Paris, France
GSI/FAIR Overview

FAIR: Facility for Antiproton and Ion Research

- Linear and Ring particle accelerators.
- Heavy Ion experiments.
- Medical irradiation facility for cancer therapy.

Green IT Cube data center:

- Measures $27 \times 30 \times 22$ meters, can hold 768 computer cabinets side by side on 6 floors.
- Highly energy-efficient, cooling with water.
Lustre at GSI

HPC at GSI/FAIR

- Green IT Cube data center
- Compute clusters
  - Prometheus (∼9000 cores, QDR IB) [decommissioned]
  - Kronos (∼10000 cores, FDR IB)
  - LCSC (∼3200 cores + ∼700 GPUs, FDR IB)
- Storage clusters
  - Hera (∼7.3PB, Lustre 1.8.9 with Debian Squeeze 2.6 Kernel)
  - Nyx (∼12PB, 70 OSSs, Lustre 2.5.3 on ZFS with Debian Wheezy 3.2 Kernel, Lustre 2.6.92 Clients on Debian Jessie 3.16 Kernel)
  - Currently in the process of moving data from Hera ⇒ Nyx, where Lustre client 1.8.9 mounts /hera (1.8.9 server) and /nyx (2.5.3 server).

GSI/FAIR is member of Intel Parallel Computing Center for developing a TSM Copytool for Lustre HSM.
Overview of Lustre HSM and Copytools

Lustre with hierarchical storage management (HSM) feature is available since Lustre version 2.5 (partially landed in Lustre 2.4), since that manifold *copytools* are developed:

- Lustre Posix Copytool
- Lustre HPSS Copytool
- Lustre S3 Copytool
- Lustre Google Drive Copytool
- Lustre Droplet Copytool
- Lustre TSM Copytool (not yet released)

Lustre HSM ≡ seamlessly *archive, release* and *restore* data.
Overview of HSM State Diagram

new file → archived → released

lfs hsm_archive

archived → dirty → archived

lfs hsm_restore

lfs hsm_release

>dd if=/dev/zero of=zeros bs=1MiB count=32 conv=sync
32+0 records in
32+0 records out
33554432 bytes (34 MB) copied, 0.401738 s, 83.5 MB/s
Overview of HSM State Diagram

- **new file**
- **archived**
- **released**
- **dirty**

- `lfs hsm_archive`
- `lfs hsm_release`
- `lfs hsm_restore`

```
> lfs hsm_state ./zeros && ll -h zeros && du -h ./zeros
./zeros: (0x00000000)
-rw-r--r-- 1 root root 32M Sep 6 13:55 zeros
32M ./zeros
```
Overview of HSM State Diagram

```
> lfs hsm_state ./.zeros && ll -h .zeros && du -h ./.zeros
./zeros: (0x00000009) exists archived, archive_id:1
-rw-r--r-- 1 root root 32M Sep 6 13:55 zeros
32M ./.zeros
```
Overview of HSM State Diagram

```
> lfs hsm_state ./zeros && ll -h zeros && du -h ./zeros
./zeros: (0x0000000d) released exists archived, archive_id:1
-rw-r--r-- 1 root root 32M Sep  6 13:55 zeros
512 ./zeros
```
Overview of HSM State Diagram

- **new file**
- **archived**
- **released**
- **dirty**

- `lfs hsm_archive` from `new file` to `archived`
- `lfs hsm_release` from `archived` to `released`
- `lfs hsm_restore` from `dirty` to `archived` and back to `dirty`

Shell command output:

```
> lfs hsm_state ./.zeros && ll -h ./.zeros && du -h ./.zeros
./zeros: (0x00000009) exists archived, archive_id:1
-rw-r--r-- 1 root root 32M Sep  6 13:55 zeros
32M ./zeros
```
Overview of HSM State Diagram

new file → archived → released

lfs hsm_archive → lfs hsm_release

lfs hsm_restore

echo 0 » ./zeros

> echo 0 >> ./zero && lfs hsm_state && ll -h zeros && du -h ./zeros

./zeros: (0x0000000b) exists dirty archived, archive_id:1
-rw-r--r-- 1 root root 33M Sep 19 09:16 ./zeros
32M ./zeros
Overview of HSM State Diagram

```
> lfs hsm_archive ./zero && lfs hsm_state && ll -h zeros && du -h ./zeros
./zeros: (0x00000009) exists archived, archive_id:1
-rw-r--r-- 1 root root 33M Sep 19 09:16 ./zeros
33M ./zeros
```
TSM Overview

Tivoli Storage Manager\(^1\) (TSM) is a client/server software from IBM employed in heterogeneous distributed environments to \textit{backup} and \textit{archive} data.

\(^1\)Now renamed to IBM Spectrum Protect.
Tape Library is a storage device consisting of:

- tape drives,
- tape cartridges,
- barcode reader,
- tape robot

GSI employs two IBM 3584-L23 Tape Libraries with an overall capacity of 1.2PB + 8.8PB.
**Backup**: A copy of the data is stored in the event the original becomes lost or damaged. Typically an incremental (forever) backup strategy is performed.

**Archive**: Remove from an on-line system those data no longer in day to day use, and place them into a long term retrievable storage (such as tape drives).

Lustre HSM is for *archiving* data.
Some TSM Features

**Compression:** Compress data stream seamlessly either on client or server side.

**Deduplication:** Eliminating duplicate copies of repeating data.

**Collocation:** Store and pack data of a client in few number of tapes as much as possible to reduce the number of media mounts and for minimizing tape drive movements.

**Storage hierarchies:** Automatically move data from faster devices to slower devices based on characteristics such as file size or storage capacity.

Meta data is stored in a DB2 database (part of TSM server).
Example Configuring Storage Hierarchies:

Example for setting data migration from small fast disk storage to slow large tape storage:

```
define devc fastdisks_devc devt=file maxcap=16G dir=/dir/dev/fastdisks
define devc slowtapes_devc devt=file maxcap=1024G dir=/dir/dev/slowtapes
define devc superslowtapes_devc devt=file maxcap=1048576G dir=/dir/dev/superslowtapes

define stg fastdisks_pool fastdisks_devc desc='fast disk storage pool' maxsize=15G nextstgpool=slowtapes_pool highmig=85 lowmig=40
define stg slowtapes_pool slowtapes_devc desc='slow tape storage pool' maxsize=1020G collocate=yes nextstgpool=superslowtapes_pool highmig=90 lowmig=50
define stg superslowtapes_pool superslowtapes_devc desc='super slow tape storage pool' maxsize=nolimit collocate=yes
```

If high threshold is reached, then move data to next storage pool hierarchy until first pool reaches the low threshold.

By means of storage hierarchies we can realized storage caching layers.

Complete installation guide for settings up a TSM server e.g. within virtual machine (KVM) is provided at http://web-docs.gsi.de/~tstibor/tsm/.
Data on TSM side is stored in an object based format.

<table>
<thead>
<tr>
<th>object # 1</th>
<th>fs: /, hl: /home/tstibor/dev/tsm/github-ltsm, ll: /README.md</th>
</tr>
</thead>
<tbody>
<tr>
<td>object id (hi,lo)</td>
<td>(0,256004)</td>
</tr>
<tr>
<td>object info length</td>
<td>32</td>
</tr>
<tr>
<td>object info size (hi,lo)</td>
<td>(0,12351)</td>
</tr>
<tr>
<td>object type</td>
<td>DSM_OBJ_FILE</td>
</tr>
<tr>
<td>object magic id</td>
<td>71147</td>
</tr>
<tr>
<td>archive description</td>
<td>readme file description</td>
</tr>
<tr>
<td>owner</td>
<td></td>
</tr>
<tr>
<td>insert date</td>
<td>2016/5/23 16:13:54</td>
</tr>
<tr>
<td>expiration date</td>
<td>2017/5/23 16:13:54</td>
</tr>
<tr>
<td>restore order (top,hi_hi,hi_lo,lo_hi,lo_lo)</td>
<td>(722,0,16813,0,0)</td>
</tr>
<tr>
<td>estimated size (hi,lo)</td>
<td>(0,12351)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>object # 2</th>
<th>fs: /, hl: /, ll: /tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>object id (hi,lo)</td>
<td>(0,256003)</td>
</tr>
<tr>
<td>object info length</td>
<td>32</td>
</tr>
<tr>
<td>object info size (hi,lo)</td>
<td>(0,86016)</td>
</tr>
<tr>
<td>object type</td>
<td>DSM_OBJ_DIRECTORY</td>
</tr>
<tr>
<td>object magic id</td>
<td>71147</td>
</tr>
<tr>
<td>archive description</td>
<td>tmp directory description</td>
</tr>
<tr>
<td>owner</td>
<td></td>
</tr>
<tr>
<td>insert date</td>
<td>2016/5/23 16:13:37</td>
</tr>
<tr>
<td>expiration date</td>
<td>2017/5/23 16:13:37</td>
</tr>
<tr>
<td>restore order (top,hi_hi,hi_lo,lo_hi,lo_lo)</td>
<td>(722,0,16812,0,0)</td>
</tr>
<tr>
<td>estimated size (hi,lo)</td>
<td>(0,86016)</td>
</tr>
</tbody>
</table>
TSM API Internals (cont.)

Taken from PDF document: Using the Application Programming Interface, Tivoli Storage Manager
typedef struct {
    unsigned int magic;
    dsStruct64_t size;
    lustre_fid fid;
} obj_info_t;

typedef struct {
    char fpath[PATH_MAX + 1];
    char desc[DSM_MAX_DESCR_LENGTH + 1];
    obj_info_t obj_info;
    dsmObjName obj_name;
} archive_info_t;

dInt16_t tsm_archive_file(const char *fs, const char *filename, const char *desc);
dInt16_t tsm_archive_fid(const char *fs, const char *filename, const char *desc, const lustre_fid *fid);
dInt16_t tsm_query_hl_ll(const char *fs, const char *hl, const char *ll, const char *desc, dsBool_t display);
dInt16_t tsm_query_file(const char *fs, const char *filename, const char *desc, dsBool_t display);
dInt16_t tsm_delete_file(const char *fs, const char *filename);
dInt16_t tsm_delete_hl_ll(const char *fs, const char *hl, const char *ll);
dInt16_t tsm_retrieve_file(const char *fs, const char *filename, const char *desc);
dInt16_t tsm_retrieve_hl_ll(const char *fs, const char *hl, const char *ll, const char *desc);
Console client ltsmc based on TSMAPI

Simple console client for testing and demonstrating

• `tsm_archive_file(const char *fs, const char *filename, const char *desc);`
• `tsm_archive_dir(const char *fs, const char *directory, const char *desc);`
• `tsm_query_hl_ll(const char *fs, const char *hl, const char *ll, const char *desc, dsBool_t display);`
• `tsm_query_file(const char *fs, const char *filename, const char *desc, dsBool_t display);
• ...

> Syntax: bin/ltsmc
-a, --archive
-r, --retrieve
-q, --query
-d, --delete
-f, --fsname <STRING>
-h, --hl <STRING>
-l, --ll <STRING>
-c, --description <STRING>
-n, --node <STRING>
-u, --username <STRING>
-p, --password <STRING>
-s, --servername <STRING>
-v, --verbose (optional level <v,vv,vvv>)
Live Demo

Demo available at http://web-docs.gsi.de/∼tstibor/tsm/ltsm-screencast-1.mp4
Summary & Outlook

• Currently in the process of hooking TSM API into llapi_hsm* functions for finalizing TSM copytool.

• TSMAPI and ltsmc is released https://github.com/tstibor/ltsm, if Lustre file system is decommissioned, then ltsm still can be used to restore data.

Thank you & questions