DE LA RECHERCHE À L'INDUSTRIE

LUSTRE/HSM BINDING IS THERE!

LAD'13 | Aurélien Degrémont <aurelien.degremont@cea.fr>

SEPTEMBER, 17th 2013

DE LA RECHERCHE À L'INDUSTRIE

cea

AGENDA

Presentation

Architecture

Components

Examples

Project status

PRESENTATION

PRESENTATION (1/3)

A long-awaited project!

- This project started several years ago.
- It has known all Lustre companies.
- After lots of modifications and rewrites, it is finally there!

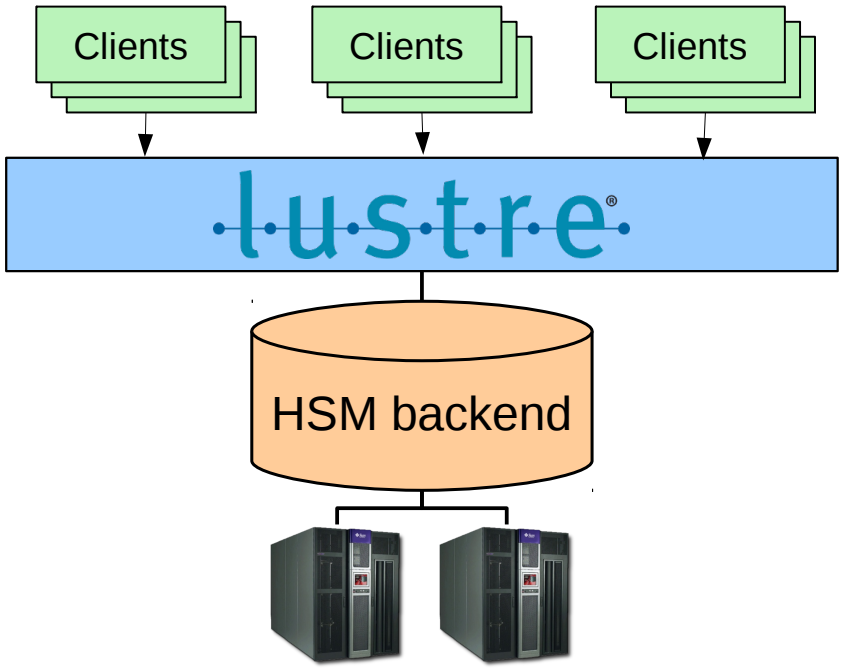
It is landed!

- Partially landed in Lustre 2.4
- Has reached total inclusion in Lustre 2.5
- Will be available in it, at the end of October 2013.

PRESENTATION (2/3)

Principle

- HSM seamless integration



- Take the best of each world:
  - **Lustre**: High performant disk-cache in front of the HSM
    - Parallel filesystem
    - High I/O performance
    - POSIX access
  - **HSM**: long term data storage
    - Manage large number of cheaper disks and tapes
    - Huge storage capacity
- Ideal for center-wide Lustre filesystem.

PRESENTATION (3/3)

Features

- Migrate data to HSM (*Archive*)
- Free disk space when needed (*Release*)
- Bring back data on cache-miss (*Restore*)

- Policy management (migration, purge, soft removal,…)
- Import from existing backend
- Disaster recovery (restore Lustre filesystem from backend)

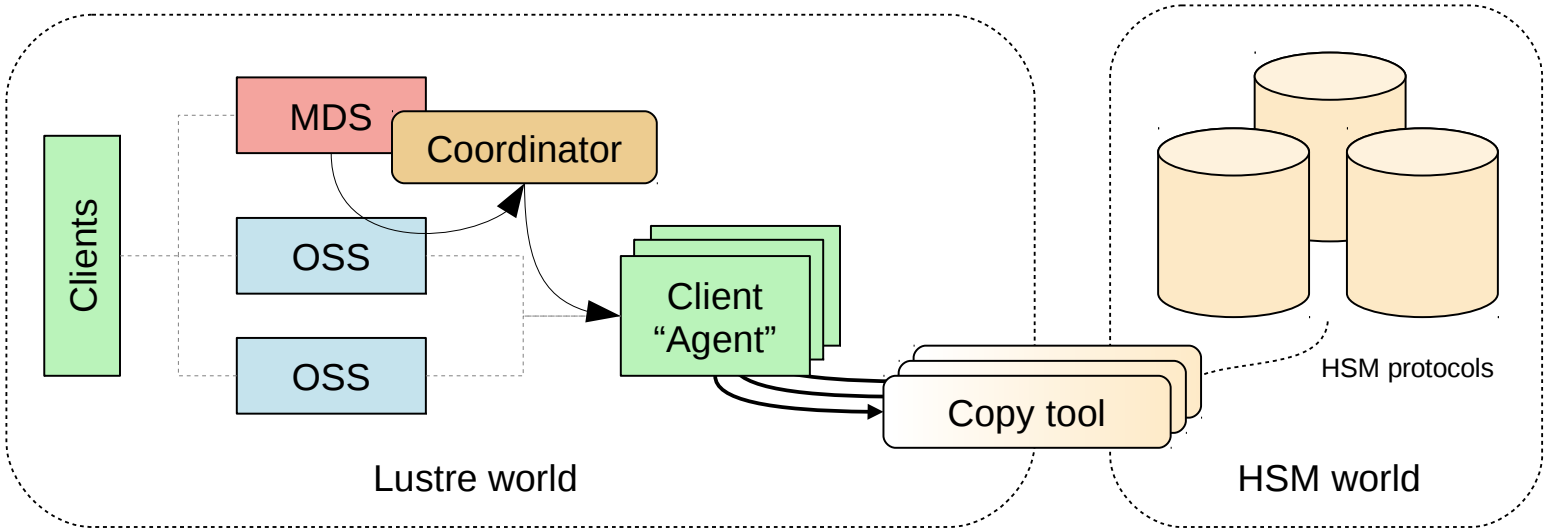
New components

- Copy tool (backend specific user-space daemon)
- Policy Engine (user-space daemon)
- Coordinator

ARCHITECTURE

ARCHITECTURE (1/2)

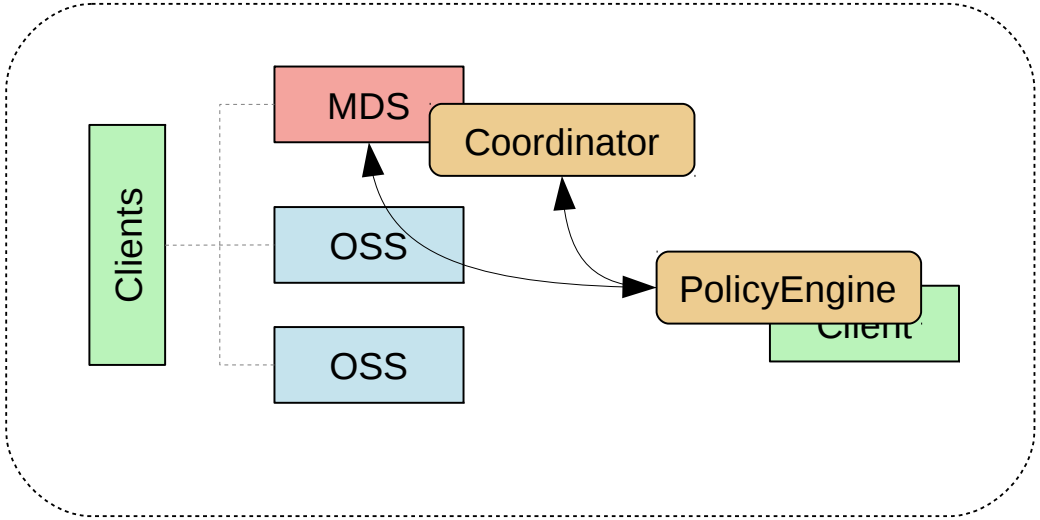
New components: *Coordinator*, *Agent* and *Copy tool*



- The coordinator gathers archive requests and dispatches them to agents.
- Agent is a client which runs a copytool to transfer data between Lustre and the HSM.

ARCHITECTURE (2/2)

PolicyEngine manages Archive and Release policies

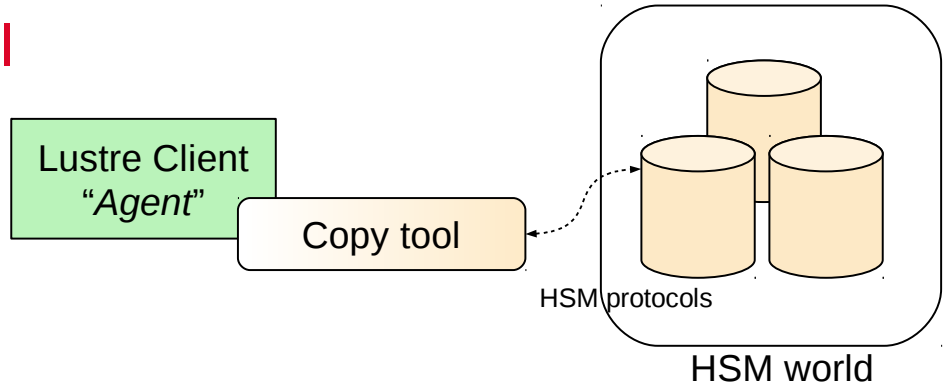


- A user-space tool which communicates with the MDT and the coordinator.
- Watches the filesystem changes.
- Triggers actions like *archive*, *release* and removal in backend.

COMPONENTS

COMPONENTS (1/4)

Copytool

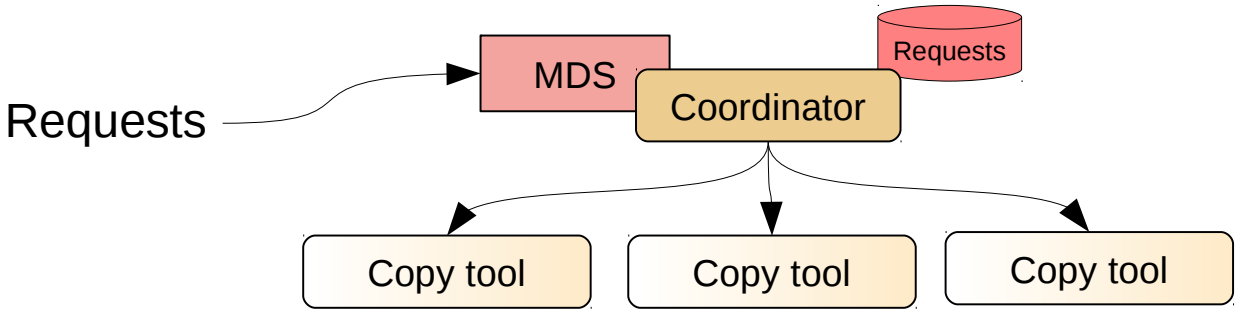


- It is the interface between Lustre and the HSM.
- It reads and writes data between them. It is HSM specific.
- It runs on a standard Lustre client (called Agent)

- 2 of them are already available:
  - **POSIX** copytool. Could be used with any system supporting a POSIX interface.
    - It is provided with Lustre
  - **HPSS** copytool. (HPSS 7.3.2+).
    - CEA development which will be freely available to all HPSS sites.
- More supported HSM to come:
  - **DMF** (SGI)
  - **OpenArchive** (GRAU DATA)

COMPONENTS (2/4)

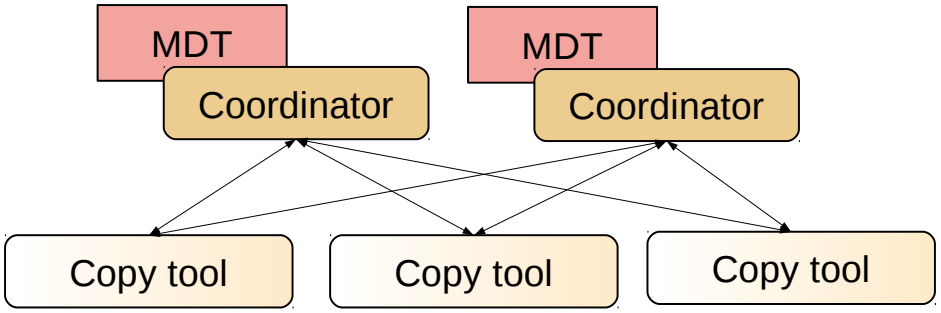
Coordinator



- MDS thread which *coordinates* HSM-related actions.
  - Centralize HSM-related requests.
  - Ignore duplicate request.
  - Control migration flow.
  - Dispatch requests to copytools.
  - Requests are saved and replayed if MDT crashes.

COMPONENTS (3/4)

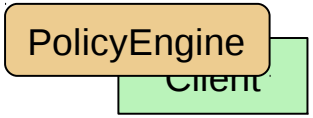
DNE compatible



- *Distributed NamespacE* feature, introduced in Lustre 2.4, is compatible with Lustre/HSM
- With the following constraints:
  - One Coordinator for each MDT
    - Each Coordinator only cares about its MDT files
  - Every copytools connect to every Coordinators
    - No cluster-wide load balancing, though
- Implementation is currently suboptimal and is to be improved in the future

COMPONENTS (4/4)

Policy Engine: RobinHood



- PolicyEngine is the specification
- RobinHood is an implementation:
  - Was first a user-space daemon for monitoring and purging large filesystems.
  - CEA opensource development (http://robinhood.sf.net)
  - Requires RobinHood 2.4.3+

Policies

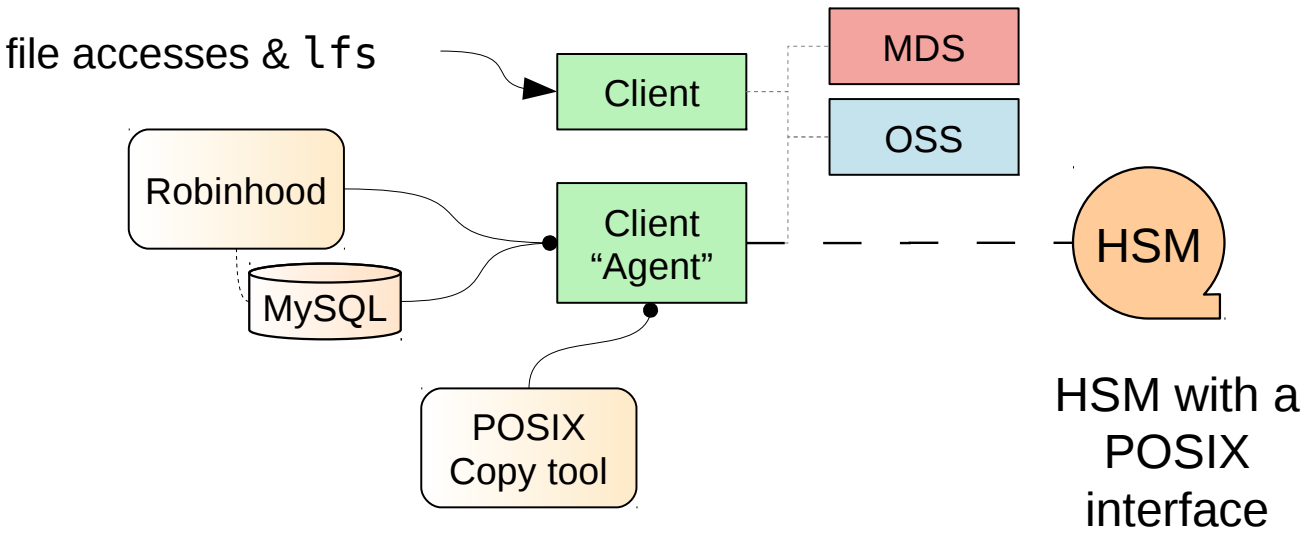
- File class definitions, associated to policies
- Based on files attributes (path, size, owner, age, xattrs…)
- Rules can be combined with boolean operators
- LRU-based migration/purge policies
- Entries can be white-listed

EXAMPLES

EXAMPLES (1/4)

Setup

- Requirements:
  - Standard Lustre v2.5 (so far, current *master* branch), sources or RPMs
  - RobinHood v2.4.3+ sources, from RobinHood website (no RPMs available yet)
- Simple configuration (theorically, 1 Lustre node is enough)



EXAMPLES (2/4)

Command line tools

- Sysadmins and users can manage file system states:

ARCHIVE

```
$ lfs hsm_archive /mnt/lustre/foo

$ lfs hsm_state /mnt/lustre/foo
/mnt/lustre/foo: (0x00000009) exists archived, archive_id:1
```

RELEASE

```
$ lfs hsm_release /mnt/lustre/foo

$ lfs hsm_state /mnt/lustre/foo
/mnt/lustre/foo: (0x0000000d) released exists archived, archive_id:1
```

AUTOMATIC RESTORE

```
$ md5sum /mnt/lustre/foo
ded5b0680e566aa024d47ac53e48cdac  /mnt/lustre/foo

$ lfs hsm_state /mnt/lustre/foo
/mnt/lustre/foo: (0x00000009) exists archived, archive_id:1
```

EXAMPLES (3/4)

Example RobinHood policy: Migration

- Migrate files older than 12 hours with a different behavior for small ones.

```
Filesets {
        FileClass small_files {
            definition { tree == "/mnt/lustre/project" and size < 1MB }
            migration_hints = "cos=12" ;
            ...
        }
}

Migration_Policies {
        ignore { size == 0 or xattr.user.no_copy == 1 }
        ignore { tree == "/mnt/lustre/logs" and name == "*.log" }

        policy migrate_small {
                target_fileclass = small_files;
                condition { last_mod > 6h or last_archive > 1d }
        }
        ...
        policy default {
                condition { last_mod > 12h }
                migration_hints = "cos=42" ;
        }
}
```

EXAMPLES (4/4)

Example RobinHood policy: Release

- Release archived files when FS usage is above 90 % but ignore some files.

```
Purge_trigger {
        trigger_on = ost_usage;
        high_watermark_pct = 90%;
        low_watermark_pct = 80%;
}

Purge_Policies {
        ignore { size < 1KB or owner == “root” }

        policy purge_quickly {
                target_fileclass = class_foo;
                condition { last_access > 1min }
        }

        ...

        policy default {
                condition { last_access > 1h }
        }
}
```

PROJECT STATUS

Client-side was landed in Lustre 2.4

- Only support compute node accesses
- No administrative task
- Does not support copytools

Full code is landed in current master branch

- Thanks to Intel, the whole code is now landed
- ETA: End of October 2013
- Will be available in Lustre 2.5, which will be the next maintenance branch

- Currently under test and debugging

Thanks.
Questions?