DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

MANAGING LUSTRE & ITS DATA @ CEA

LUG Japan | Aurelien Degremont <aurelien.degremont@cea.fr>
CEA, DAM, DIF, F-91297 ARPAJON CEDEX

October 17, 2013

AGENDA

WHAT IS CEA?

LUSTRE ARCHITECTURE

LUSTRE DEVELOPMENTS

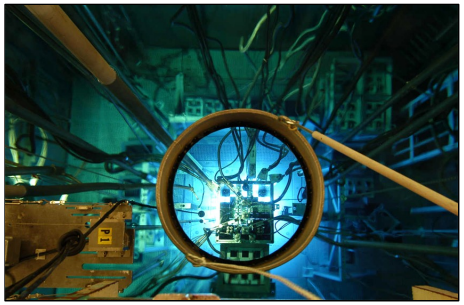
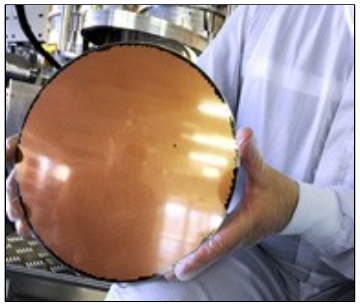
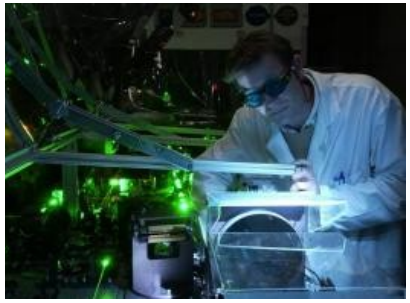
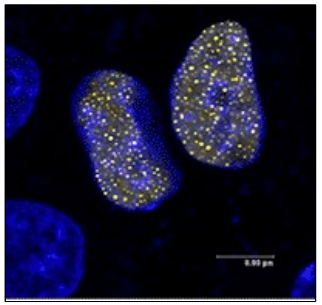
- SHINE

- ROBINHOOD

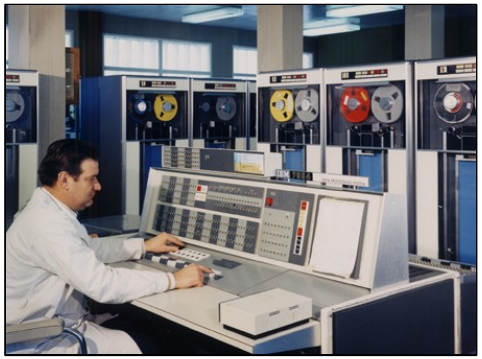
- LUSTRE/HSM BINDING

WHAT IS CEA?

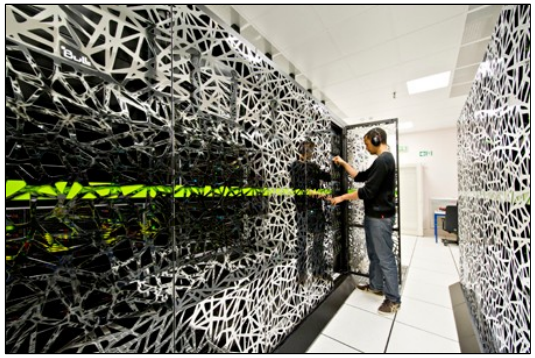
- CEA, Commissariat à l'Energie Atomique et aux Energies Alternatives
  French agency dedicated to research in energy, physics, biology, electronics fields, ...

- CEA is running supercomputers for decades for science simulations.



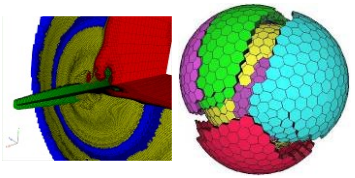
1963



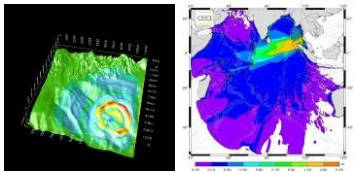
2012

EVOLUTION OF CEA COMPUTING COMPLEXES

Deterrence



Security



2012

CCRT-C

2011

CURIE

2010

TERA-100

2007

CCRT-B

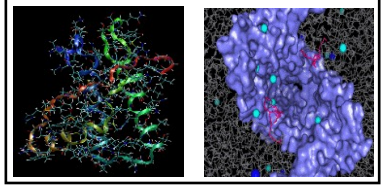
2005

TERA-10

2003

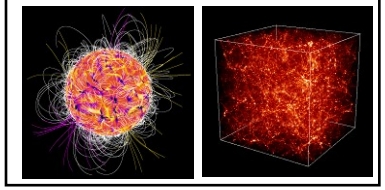
CCRT

Biology

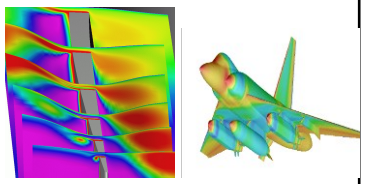


lustre®

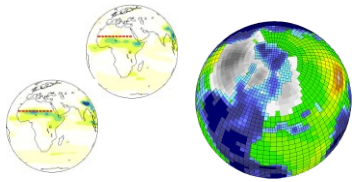
Astrophysics



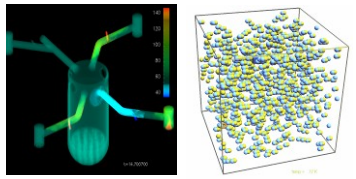
Aeronautics



Climate



Nuclear Energy



LUSTRE ARCHITECTURE

COMPUTING CENTERS

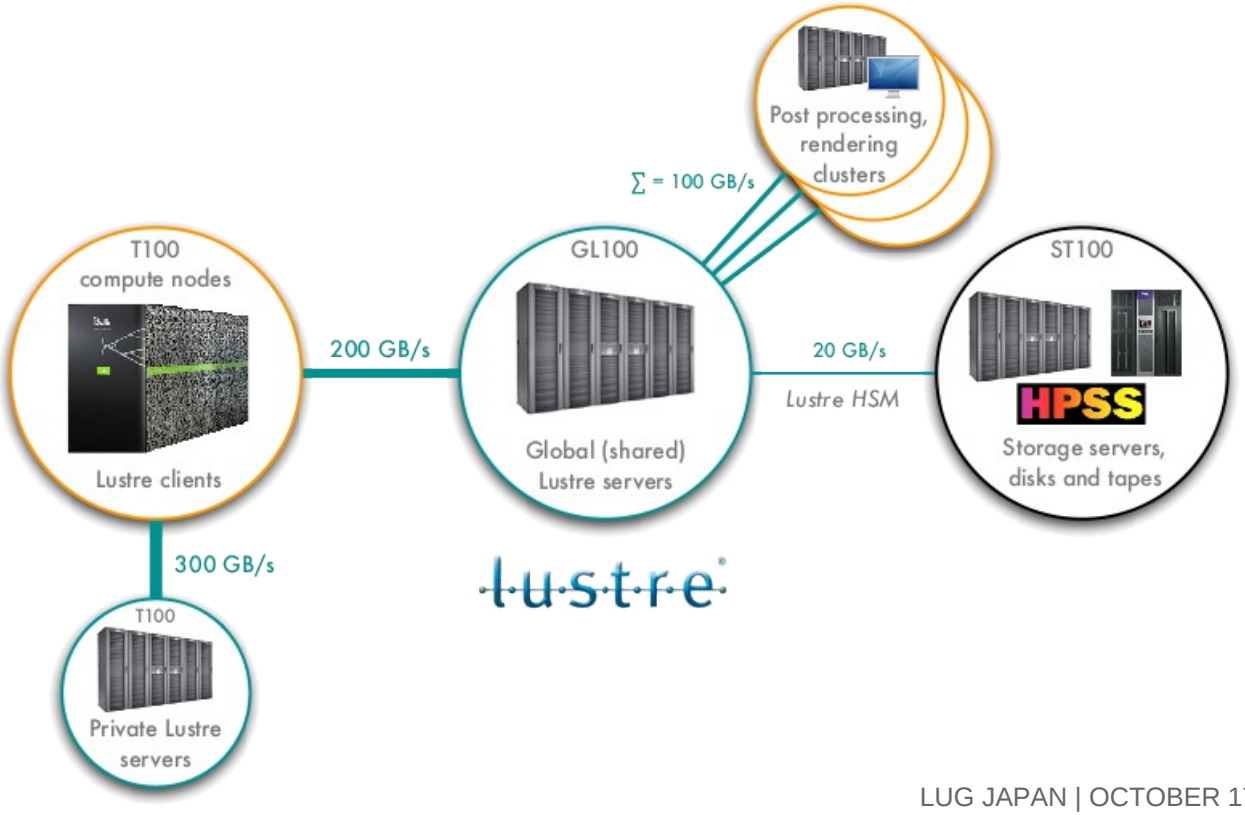
3 computing centers in few numbers

- Lustre is well known to CEA teams since for 10 years now.
- It is used in production on clusters since 2003.
- Moving to dedicated scratch filesystems to complex, center-wide, data management relying on a full Lustre infrastructure.

- TERA
  - 3 supercomputers
  - More than 1 PFlops
  - More than 5000 Lustre nodes
  - 500 GB/s of Lustre bandwidth
  - 23+ PB of Lustre filesystems
  - 12 Lustre filesystems
- TGCC/CCRT
  - 2 supercomputers
  - More than 6200 Lustre nodes
  - 250 GB/s of total Lustre bandwidth
  - 12+ PB of Lustre filesystems
  - 5 Lustre filesystems

DATA-CENTRIC LUSTRE ARCHITECTURE

- Scratch data are local to clusters
- Simulation results are directly written to a central filesystem
  - Zero copy data access for post-processing clusters
  - Directly connected to HSM
  - Automatic and transparent migration between Lustre and HSM
  - Lustre as a very big cache in front of the HSM



LUSTRE DEVELOPMENTS

SHINE

SHINE: SCALABLE LUSTRE MANAGEMENT

Shine is an open source Python-based tool

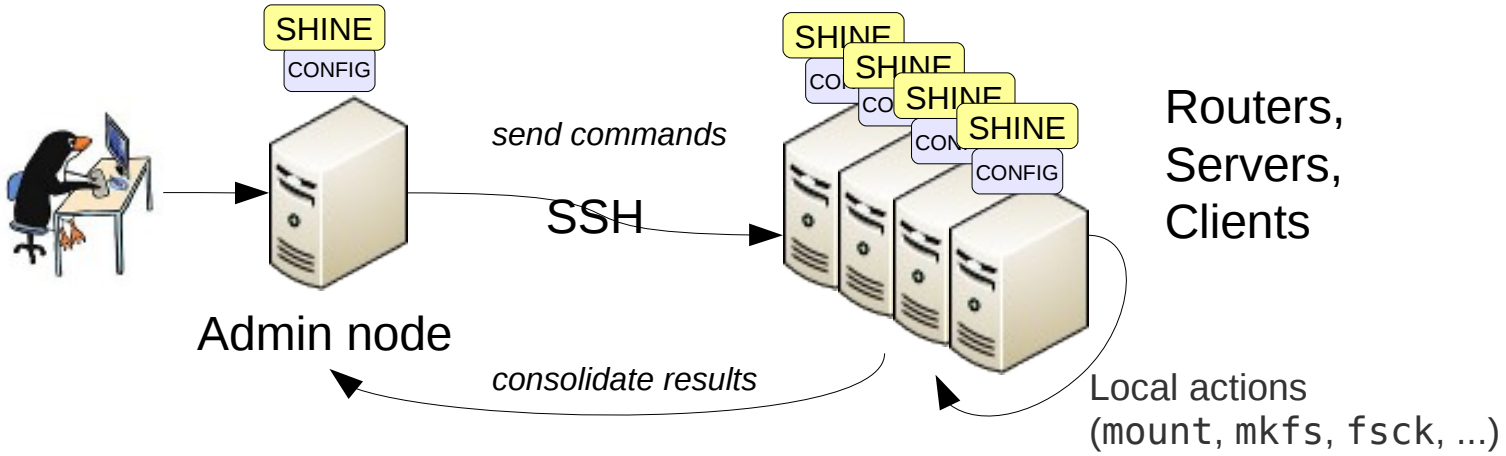
- CEA open source project, since 2007, in Python (v2.4+).
- Latest version 1.3, freely available on SourceForge :
  - `http://lustre-shine.sf.net/`

- Objectives :
  - Hide Lustre complexity. Do not need to be a Lustre expert to administrate it!
    - Throw away all your Lustre wrapper scripts
  - Highly scalable to meet big CEA Lustre filesystem constraints.
    - Rely on ClusterShell for efficient command execution
    - Run on 5000 clients and 1000 OSTs in few seconds

- Based on a CLI (admins like working with terminals)
  - Customizable
  - Aggregate results and display consolidated outputs

- Validated on Lustre version 1.8 to 2.4

SHINE: ARCHITECTURE



- Setup
  - Shine is deployed on management and all Lustre nodes (Only 2 RPMs)
  - Shine heavily relies on your existing SSH infrastructure.
    - No complex communication daemons! No key! No additional config files!
  - Shine replicates filesystem configuration on all filesystem nodes

- Interface
  - Admins control the filesystem through a central point of management
    - Shine will connect to required nodes transparently
  - Or run locally on remote node for local actions only.

SHINE: CONFIGURATION FILE

Model file

- Lustre filesystem components are described in a configuration file called a *model*.
- This model should include:
  - File system name

```
fs_name: tokyo
```

  - NID/node mapping

```
nid_map: nodes=nova[2-5] nids=nova[2-5]@tcp0
```

  - Device per target type

```
# MGS
mgt: node=nova2 dev=/dev/sde1

# MDT
mdt: node=nova3 dev=/dev/sdf

# OST
ost: node=nova4 ha_node=nova5 dev=/dev/mapper/lun[1-6]
ost: node=nova5 ha_node=nova4 dev=/dev/mapper/lun[6-11]
```

  - Clients and mount path

```
client: node=nova[10-19]
mount_path: /mnt/lad2012
```

- And that's sufficient!

SHINE: USAGE EXAMPLES

- Format
  - No issue with MGS NIDs or failover NIDs.

```
# shine format -f tokyo
Format tokyo on nova[2-5]: are you sure? (y)es/(N)o: y
Starting format of 14 targets on nova[2-5]
 FILESYSTEM STATUS (tokyo)
TYPE  # STATUS  NODES
----  - ------  -----
MGT   1 offline nova2
MDT   1 offline nova3
OST  12 offline nova[4-5]
```

- Status
  - With a recovery in progress on 1 OST for 5 clients

```
# shine status -f tokyo -x sicknode
 FILESYSTEM STATUS (tokyo)
TYPE  # STATUS                   NODES
----  - ------                   -----
MGT   1 online                   nova2
MDT   1 online                   nova3
OST  11 online                   nova[4-5]
OST   1 recovering for 99s (0/5) nova5
CLI   5 mounted (recovering=1)   nova[10-14]
CLI   5 mounted                  nova[15-19]
```

SHINE: FULLY FEATURED

Lots of other features not detailed here

- Display
  - Consolidate views
  - High control on display
- Lustre components
  - Routers support (start, stop and status)
  - Client-only or MGS-only filesystems
- Lustre tunings and configurations
  - External journal device
  - Default striping
  - Format options
  - Mount options
  - Mount path
  - Multirail: Multiple NIDs per server
  - Eviction detections
  - Quota
  - Tunefs
- And more...

LUSTRE DEVELOPMENTS

ROBINHOOD

ROBINHOOD

Swiss army knife for your filesystem

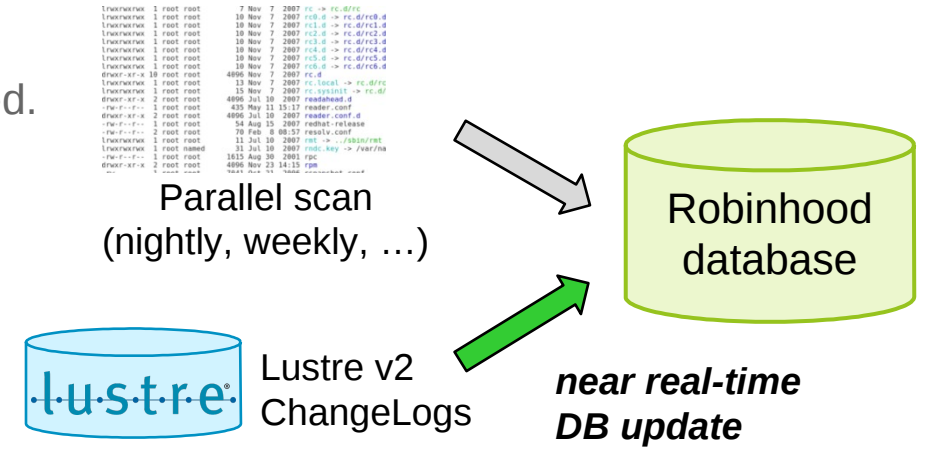
- Policy Engine and reporting tool for large filesystem
- Open Source product developped by CEA
  - `http://robinhood.sf.net/`
- Current version is 2.4.3, but next release, 2.5, adds new features and performances.

- Monitor filesystem activities by scanning or reading events (Lustre Changelogs)
- Save all metadata in a (My)SQL database
- Query this database at will to:
  - Audit, accounting, alerts
  - Migration, purge
  - Everything based on policies
- Thanks to CLI or WebGUI

- Use to control good usages and keep your filesystem healthy

Feeding the database

- Robinhood information and actions are based upon the database data.
  - Robinhood supports MySQL as backend.
- Database could be filled using:
  - Parallel filesystem scan
    - For Lustre 1.8 or any POSIX filesystem.
  - Reading Lustre Changelog
    - For Lustre 2.x
    - Only an initial scan is needed.



ROBINHOOD: RBH-FIND, RBH-DU

Fast *find* and *du* clones

- Query Robinhood DB instead of performing POSIX namespace scan
  ➔ faster!

```
> rbh-find [path] -user “foo*" -size +1G –ost 4
```

20sec for 40M entries

- Enhanced *du* :
  - Detailed stats (by type…)
  - Can filter by user

```
> rbh-du -sH /fs/dir -u foo --details /fs/dir
    symlink count:30777,   size:1.0M, spc_used:9.1M
    dir     count:598024,  size:2.4G, spc_used:2.4G
    file    count:3093601, size:3.2T, spc_used:2.9T
```

ROBINHOOD: FINE-GRAINED STATISTICS

Top users and groups

- Sorted by volume, object count, avg file size…

```
> rbh-report --top-users --by-count
rank, user      ,   spc_used,      count,   avg_size
   1, john      ,  423.23 GB,    1599881,  275.30 KB
   2, paul      ,  292.91 GB,     954153,  330.98 KB
   3, mike      ,   65.37 GB,     543169,  130.98 KB
…
```

Top directories

- Sorted by object count, avg file size…

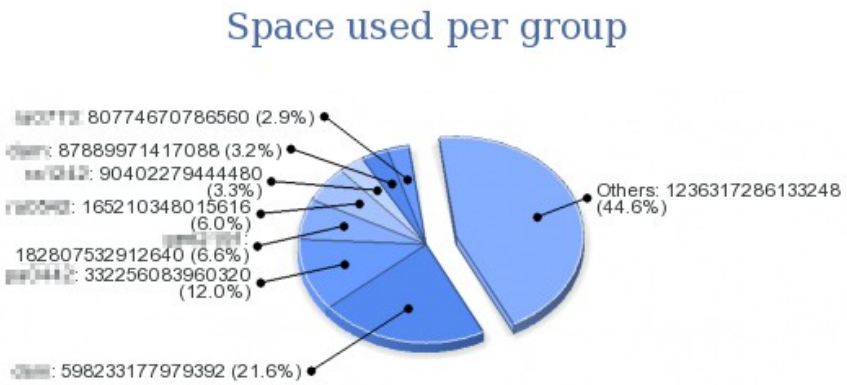
```
> rbh-report --top-dirs --by-count
rank,            path, dircount,   avgsize, user, group,  last_mod
   1, /hpss/foo1/dir1,    24832,   2.62 GB, foo1, gr59,  2013/03/11 17:13:45
   2, /hpss/foo2/dir3,    20484, 339.88 MB, foo2,  g03,  2013/02/03 06:59:05
   3, /hpss/bar2/dir4,    19484, 543.82 MB, bar2,  g03,  2012/05/28 12:45:26
…
```

ROBINHOOD : WEBGUI

Web GUI

- Users
- Groups
- Sizes
- Search

Volume | Count | List

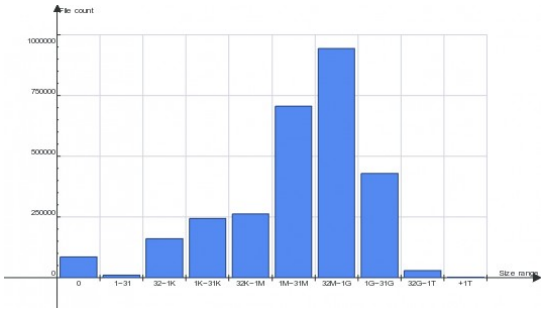


Group	Space used	Count
ra1	544.09 TB	12 708 782
pa0	302.19 TB	2 743 617
gen2	166.26 TB	3 370 537
ra05	150.26 TB	1 995 581
ra12	82.22 TB	1 393 682
ra2	79.94 TB	7 465 108
ra07	73.46 TB	522 402

File size profiling

- Available in the Robinhood web interface
- File size repartition

Global / per file size



Summary per user

User	Total volume	File count	Avg file size	file size ratio			
				empty	<1K	<1M	<1G
	11.98 TB	36 251	346.48 MB	0.74%	3.17%	67.12%	97.65%
	15.02 TB	86 584	181.94 MB	21.73%	36.00%	86.33%	93.63%
	120.02 TB	114 445	1.07 GB	13.30%	30.75%	65.83%	75.80%
	27.31 TB	65 944	434.31 MB	21.16%	30.35%	75.72%	84.30%
	7.53 TB	27 725	284.86 MB	0.76%	26.44%	80.27%	90.80%
	19.17 TB	87 617	229.41 MB	9.19%	22.32%	61.84%	81.17%
	74.17 TB	38 685	1.96 GB	16.62%	22.65%	50.07%	60.83%
	57.35 TB	112 730	533.41 MB	1.97%	6.21%	36.07%	65.95%

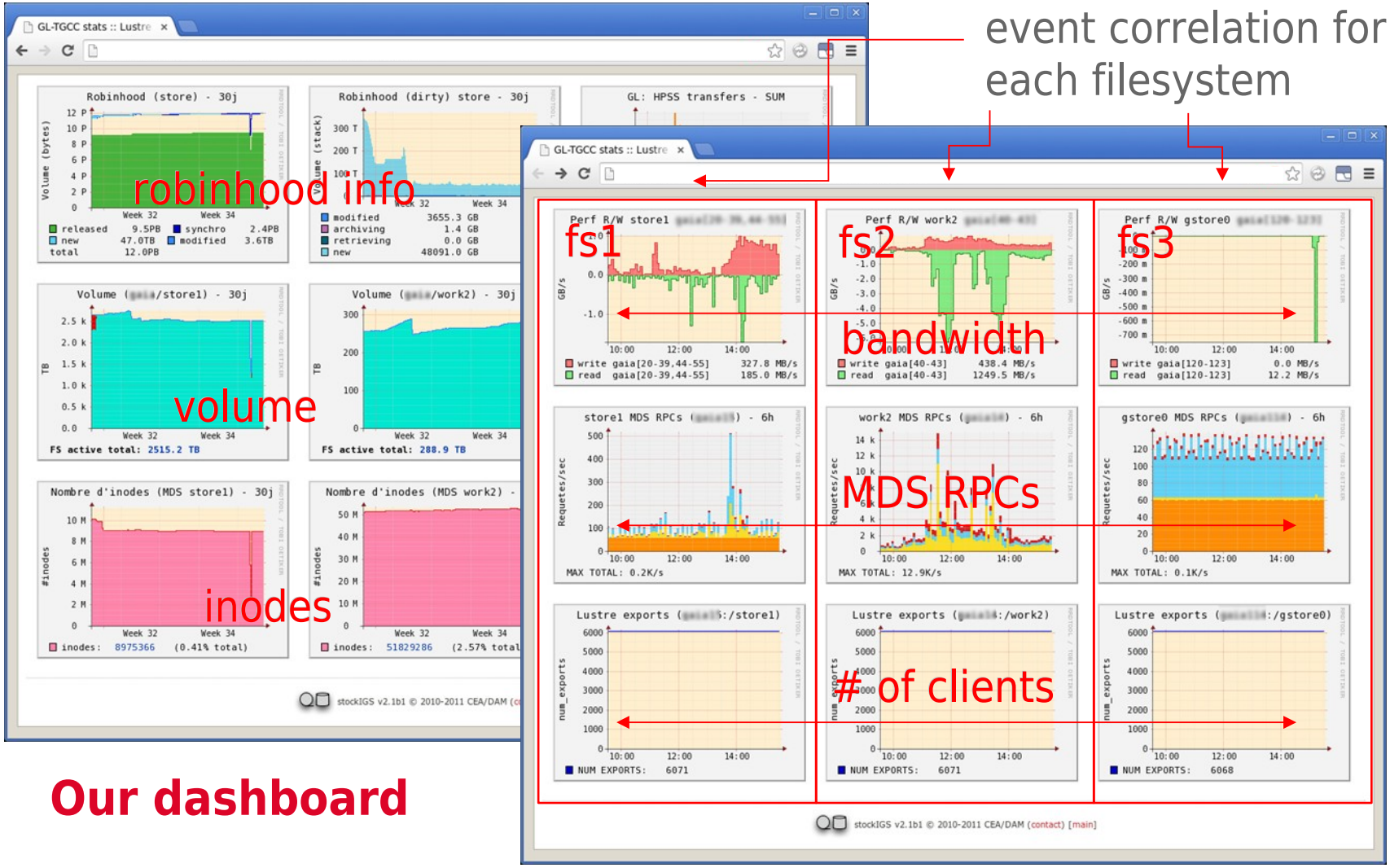
User guidance

- Informational emails sent on specific criteria

Avg. file size < 50MB, more than 80% of files < 32MB
Includes good I/O practices, stats about their account (list of non-compliant directories)

USING ROBINHOOD FOR HOME-MADE GRAPHS

column view allows event correlation for each filesystem



Our dashboard

FS TEMPERATURE: USER WORKING SETS

Time-lapse of filesystem usage

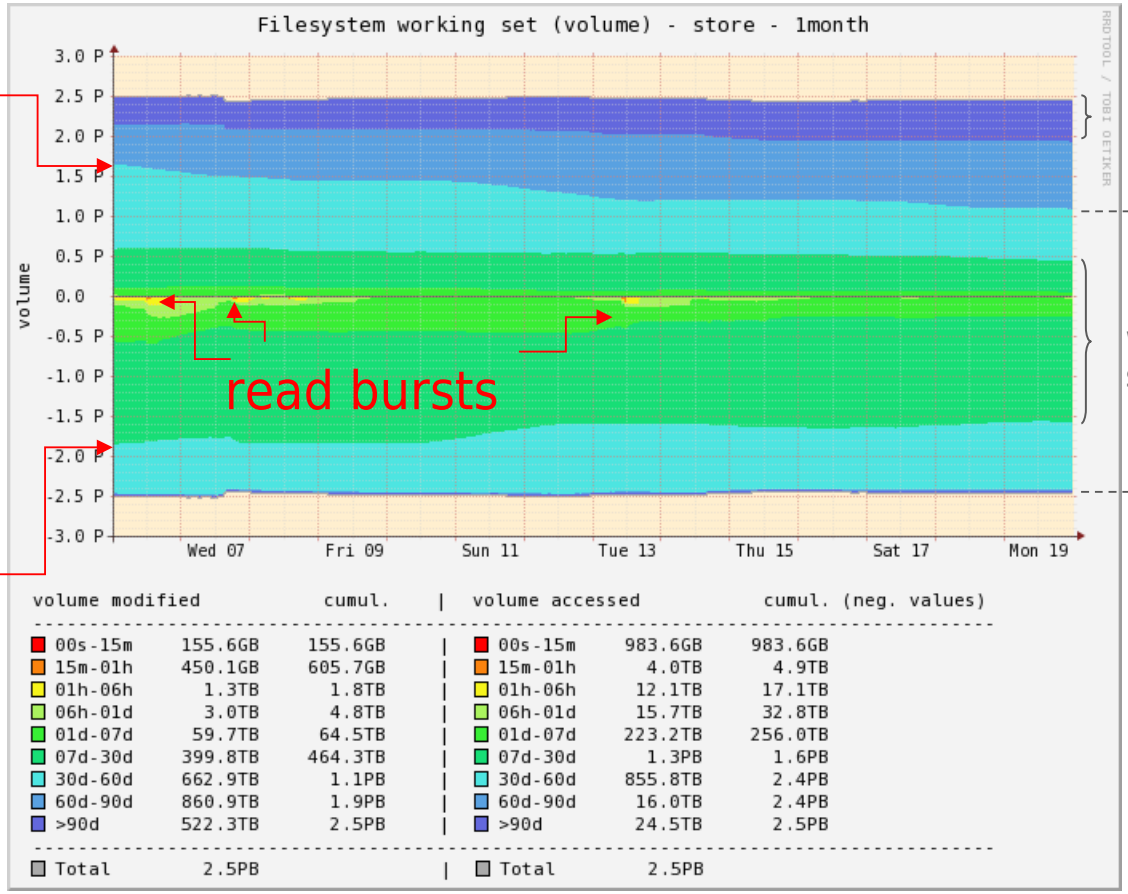
- working set = set of files *recently* written/read

70% of data produced within the last 2 months

data **production** (mod. time)

data **in use** (last access)

80% of data accessed <1 month



volume modified		cumul.	volume accessed		cumul. (neg. values)
00s-15m	155.6GB	155.6GB	00s-15m	983.6GB	983.6GB
15m-01h	450.1GB	605.7GB	15m-01h	4.0TB	4.9TB
01h-06h	1.3TB	1.8TB	01h-06h	12.1TB	17.1TB
06h-01d	3.0TB	4.8TB	06h-01d	15.7TB	32.8TB
01d-07d	59.7TB	64.5TB	01d-07d	223.2TB	256.0TB
07d-30d	399.8TB	464.3TB	07d-30d	1.3PB	1.6PB
30d-60d	662.9TB	1.1PB	30d-60d	855.8TB	2.4PB
60d-90d	860.9TB	1.9PB	60d-90d	16.0TB	2.4PB
>90d	522.3TB	2.5PB	>90d	24.5TB	2.5PB
Total	2.5PB		Total	2.5PB	

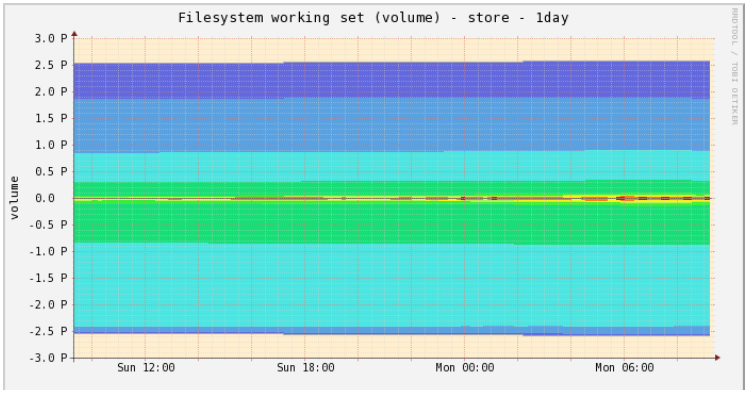
"Cold" data

1 month working set

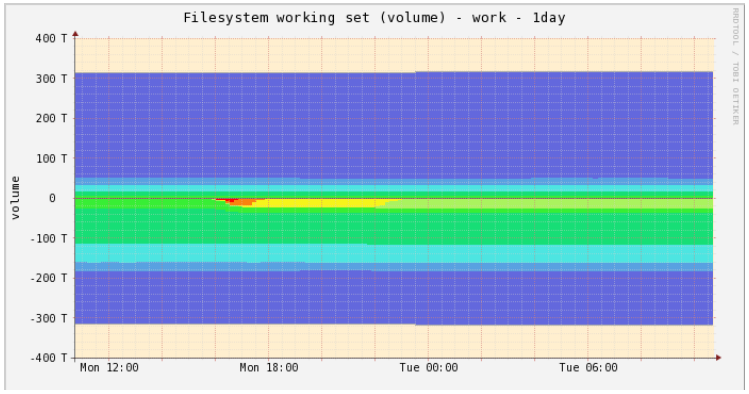
2 months working set

FS TEMPERATURE: EXAMPLES

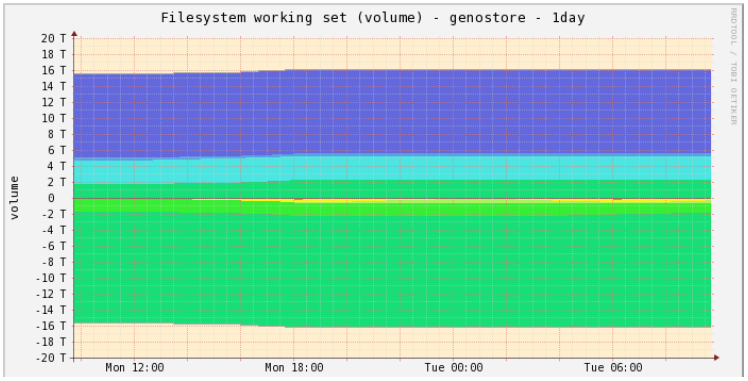
Visualization of different filesystem usage patterns



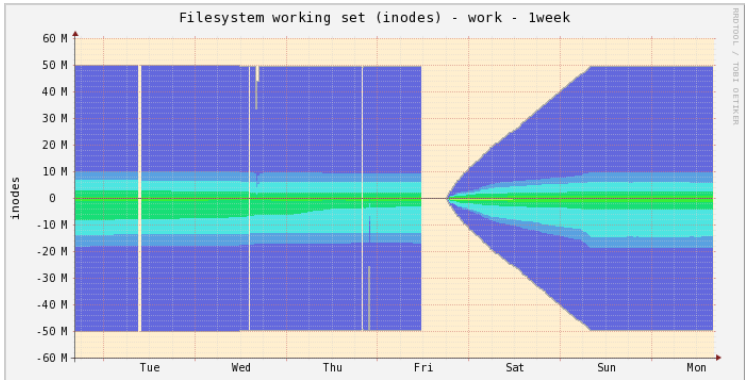
significant reads/writes (volume)



cooling effect after a large read (volume)



read-mostly filesystem (volume)



Robinhood DB dump and initial scan (inodes)

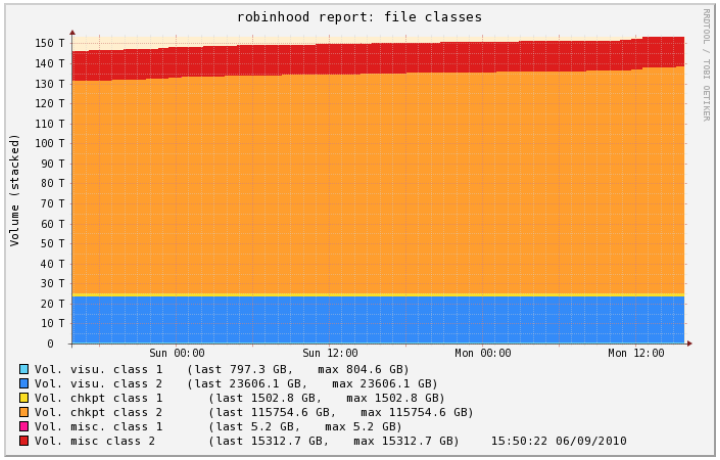
nice linear scan, ~1.5 days for 50M inodes

ROBINHOOD POLICIES

Fileclasses based on file attributes

- Admin defined rules
- Policy definition:
  - Flexible and highly customizable
  - Attribute-based
  - Using fileclass definitions

```
Fileclass system_log_files {
    definition { name == “*.log” and
                 (owner == “root” or group == “root”) }
}
Fileclass big_pictures {
    definition { (name == “*.png” or name == “*.jpg”)
                  and (size > 10MB) }
}
Fileclass flagged {
    definition { xattr.user.flag == “expected value” }
}
```



- Get class summary

```
$ rbh-report --classinfo
class             ,    count,   spc_used,     volume,  min_size,   max_size,   avg_size
Documents         ,   128965,  227.35 TB,  250.69 TB,   8.00 KB,    2.00 TB,   30.03 GB
System_log_files  ,     1536,    4.06 TB,    4.06 TB,       684,  200.01 GB,    2.71 GB
Big_pictures      ,   621623,  637.99 TB,  638.02 TB,         3,    1.54 TB,    1.05 GB
```

Apply policies to fileclasses

- Several built-in policies
  - Purge (for scratch filesystem)
  - Directory removal
  - Deferred removal (for undelete)
  - Backup and Archiving
  - HSM: schedule *archiving* and *release*

- Examples:

```
fileclass BigLogFiles {
    definition {
        type == file and size > 100MB
          and (path == /fs/logdir/* or name == *.log)
    }
     …
}
```

```
purge_policies {
    ignore_fileclass = my_fileclass;

    policy purge_logs {
         target_fileclass = BigLogFiles;
         condition { last_mod > 15d }
    }
}
```

LUSTRE DEVELOPMENTS

LUSTRE/HSM BINDING

LUSTRE/HSM

A long-awaited project!

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

It is landed!

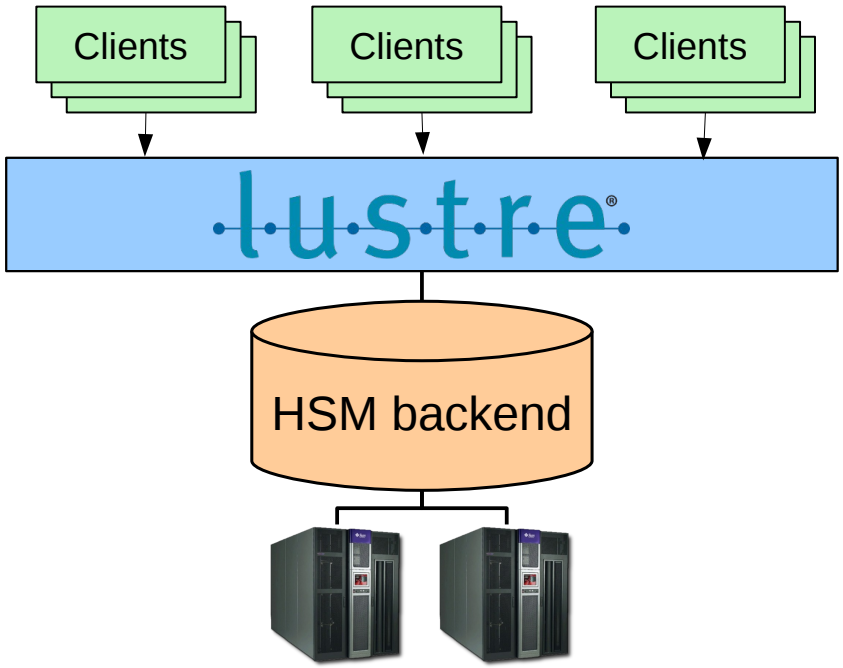
- Thanks to Intel, the whole code is now landed
- Partially landed in Lustre 2.4
- Has reached total inclusion in Lustre 2.5
- Will be available in it, at the end of October 2013, which will be the next maintenance branch

- Currently under test and debugging

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*)

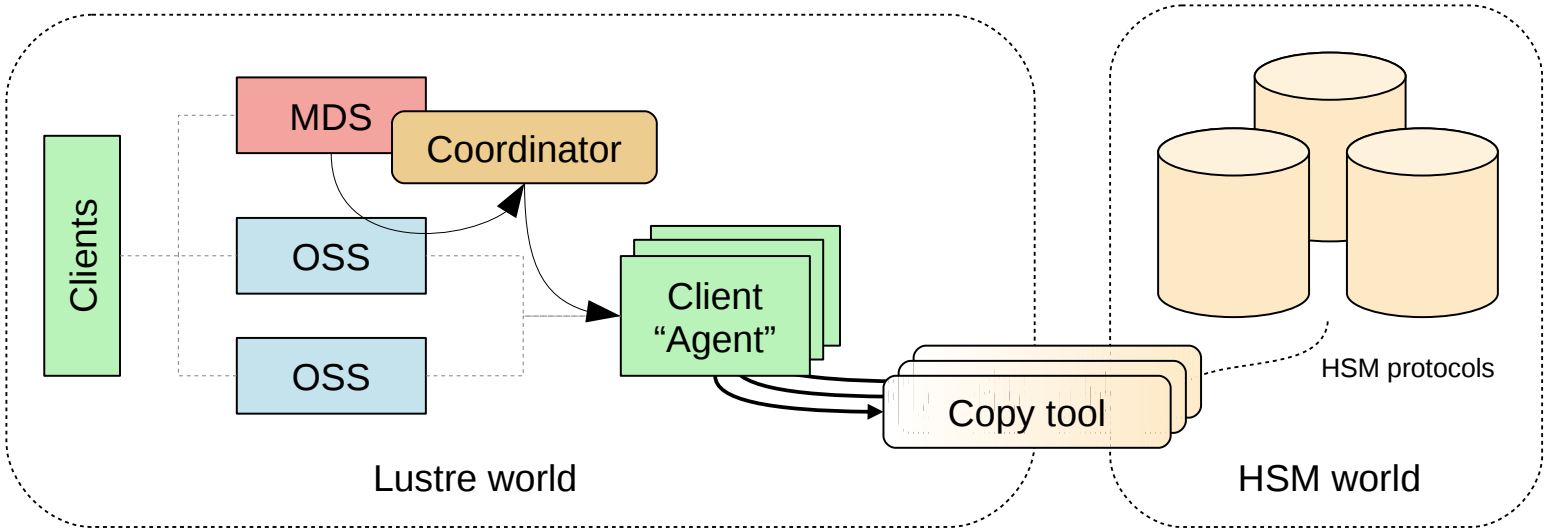
- Policy management (migration, purge, soft removal,…)
- Import from existing backend
- Disaster recovery (restore Lustre filesystem from backend)

New components

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

ARCHITECTURE (1/2)

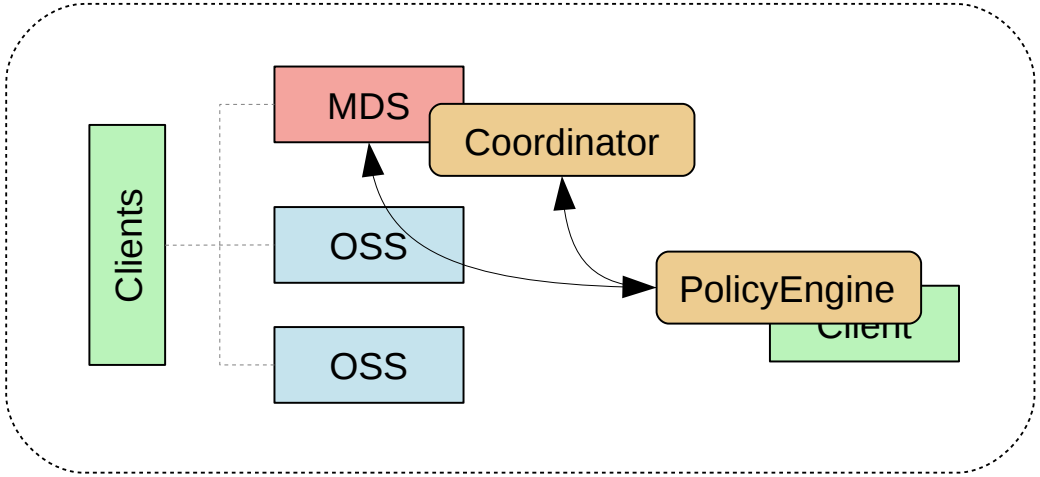
New components: *Coordinator*, *Agent* and *Copy tool*



- 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)

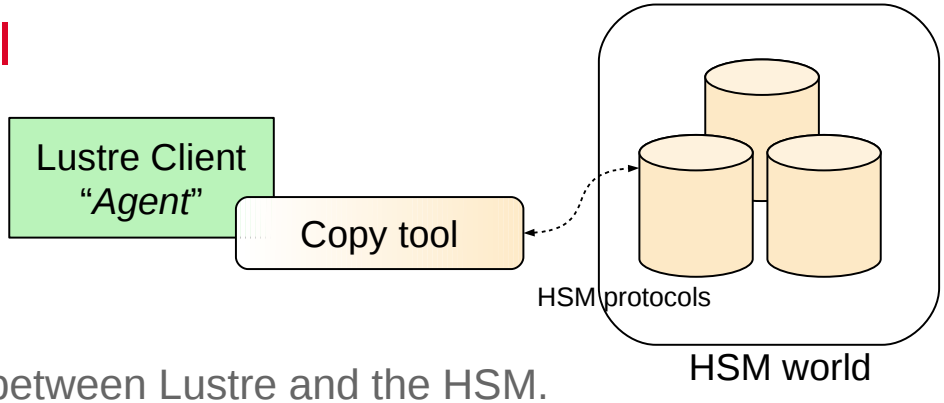
PolicyEngine manages Archive and Release policies



- 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

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)

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" ;
        }
}
```

CONCLUSION

CONCLUSION

- CEA has a long history of Lustre usage.
- It has developed a deep knowledge of it
  - This is useful for a good system administration
  - And helps to develop tools and patches for Lustre

- File system patterns and volumes are closely watched.
- Tools are developed to :
  - Watch filesystem usages
  - Advice user of better pratices
  - Remove and migrate data to optimize filesystem usage
  - Understand real user needs and estimate future storage needs

Thanks.
Questions?

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex

DAM
DSSI

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019