RobinHood Policy Engine

`http://robinhood.sf.net`

European Lustre Workshop 2011

September 26-27, 2011

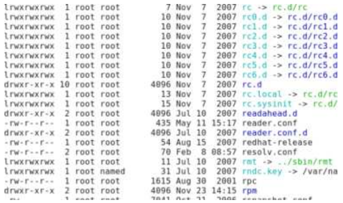
Thomas LEIBOVICI
thomas.leibovici@cea.fr

Robinhood in a Nutshell

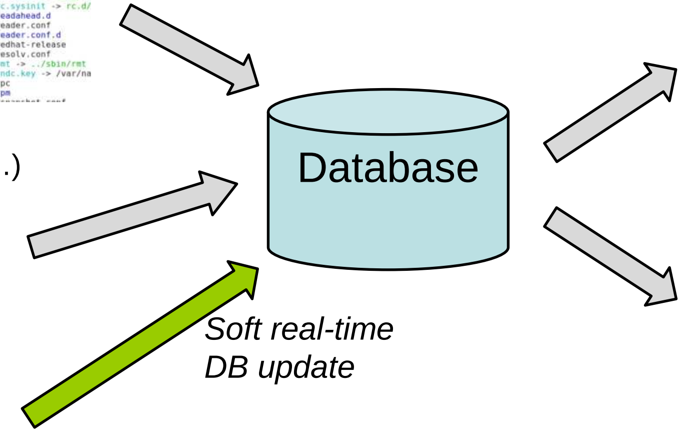
- **Robinhood is a PolicyEngine**
  - Apply various policies on any POSIX filesystem
  - Policy rules based on:
    - file properties (path, owner, size, modification/access time, …)
    - xattr values
    - Lustre specific: ost_index, …
  - Entries can be whitelisted
  - Extra capabilities for Lustre:
    - OST aware
    - Pool aware
    - Lustre 2.x :
      - handle entries by fid
      - process MDT ChangeLogs (no scan needed)
  - Accounting, reporting
- **Massively multi-threaded**
  - // scan, // purge, …
- **OpenSource**
  - CeCILL-C: LGPLv3 compatible

RobinHood: Big Picture

- **Principle: scan sometimes (…or never with Lustre v2) query often**



```
lrwxrwxrwx  1 root root        7 Nov  7  2007 rc -> rc.d/rc
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc0.d -> rc.d/rc0.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc1.d -> rc.d/rc1.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc2.d -> rc.d/rc2.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc3.d -> rc.d/rc3.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc4.d -> rc.d/rc4.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc5.d -> rc.d/rc5.d
lrwxrwxrwx  1 root root       10 Nov  7  2007 rc6.d -> rc.d/rc6.d
drwxr-xr-x 10 root root     4096 Nov  7  2007 rc.d
lrwxrwxrwx  1 root root       13 Nov  7  2007 rc.local -> rc.d/rc
lrwxrwxrwx  1 root root       15 Nov  7  2007 rc.sysinit -> rc.d/
drwxr-xr-x  2 root root     4096 Jul 10  2007 readahead.d
-rw-r--r--  1 root root      435 May 11 15:17 reader.conf
drwxr-xr-x  2 root root     4096 Jul 10  2007 reader.conf.d
-rw-r--r--  1 root root       54 Aug 15  2007 redhat-release
-rw-r--r--  2 root root       70 Feb  8 08:57 resolv.conf
lrwxrwxrwx  1 root root       11 Jul 10  2007 rmt -> ../sbin/rmt
lrwxrwxrwx  1 root named      31 Jul 10  2007 rndc.key -> /var/na
-rw-r--r--  1 root root     1615 Aug 30  2001 rpc
drwxr-xr-x  2 root root     4096 Nov 23 14:15 rpm
```



Regular scan
(nighly, weekly, …)

FS dump*
(*robinhood 2.4)

Lustre v2
ChangeLogs

Soft real-time
DB update

- Robinhood querying tool
- SQL

Build-in features / policies:
- purge files by LRU
- customizable alerts
- quota alerts
- « soft rm »
…

- Info is always available in DB when needed
- Flexible SQL querying (filters, sort, group, …)
- Searches do not load the filesystem
- DB schema can be optimized for fast customized accounting
- Soft real-time DB update with Lustre v2 Changelogs

Common Usages

Common usages of Robinhood PE:

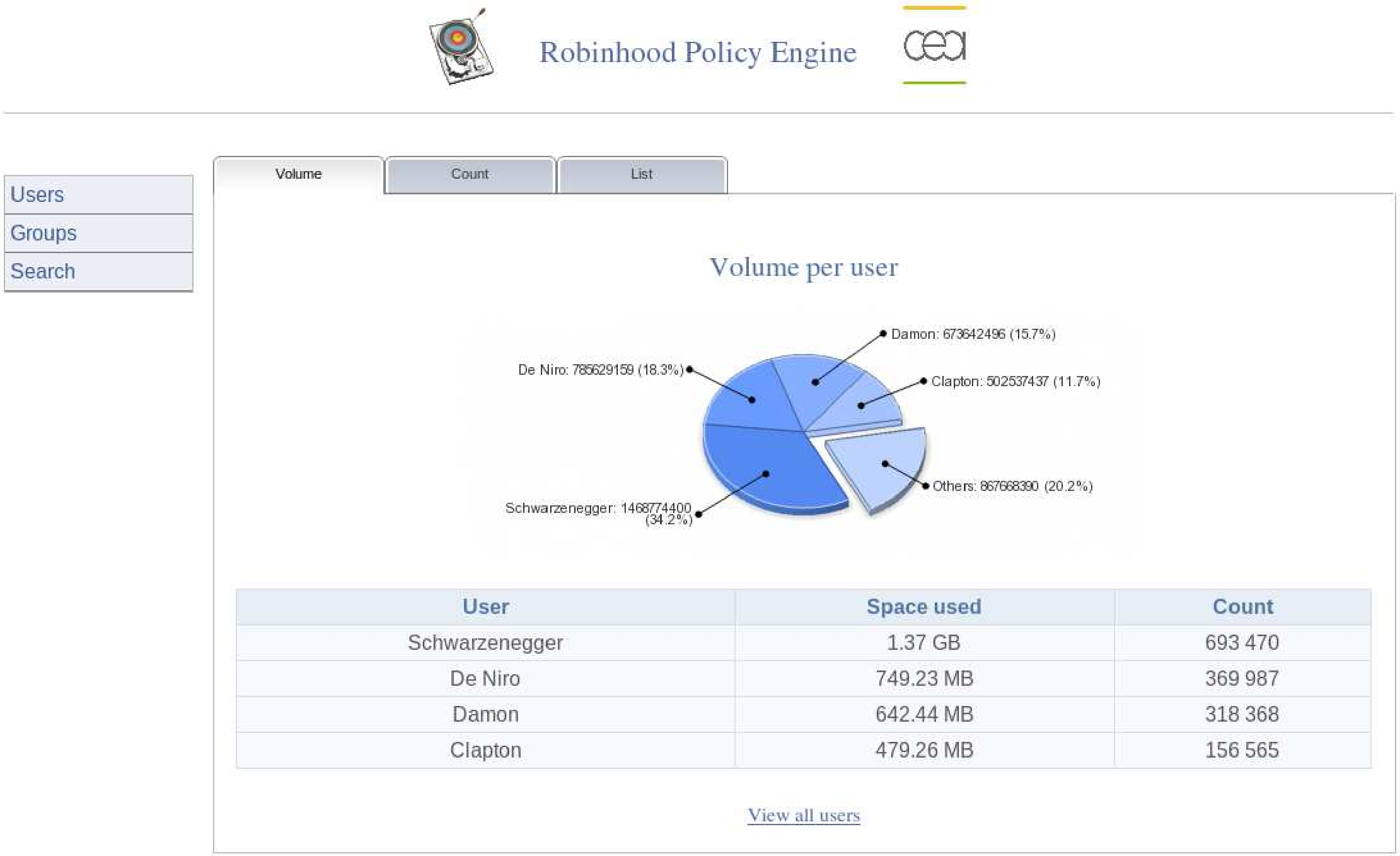
- **Statistics, accounting and monitoring**
  - Stats
  - Alerts
- **Managing a scratch filesystem**
  - Clean old files / manage file life-time
  - Remove whole directories according to policy rules
- **Data archiving and HSM binding**
  - Archiving policy
  - Purge policy (release disk space)
  - Deferred removal policy (un-delete)

Use case 1

Statistics, Accounting and Monitoring

Web interface

- **Web interface to visualize usage per user / per group**



Robinhood Policy Engine

Users
Groups
Search
Volume
Count
List

Volume per user

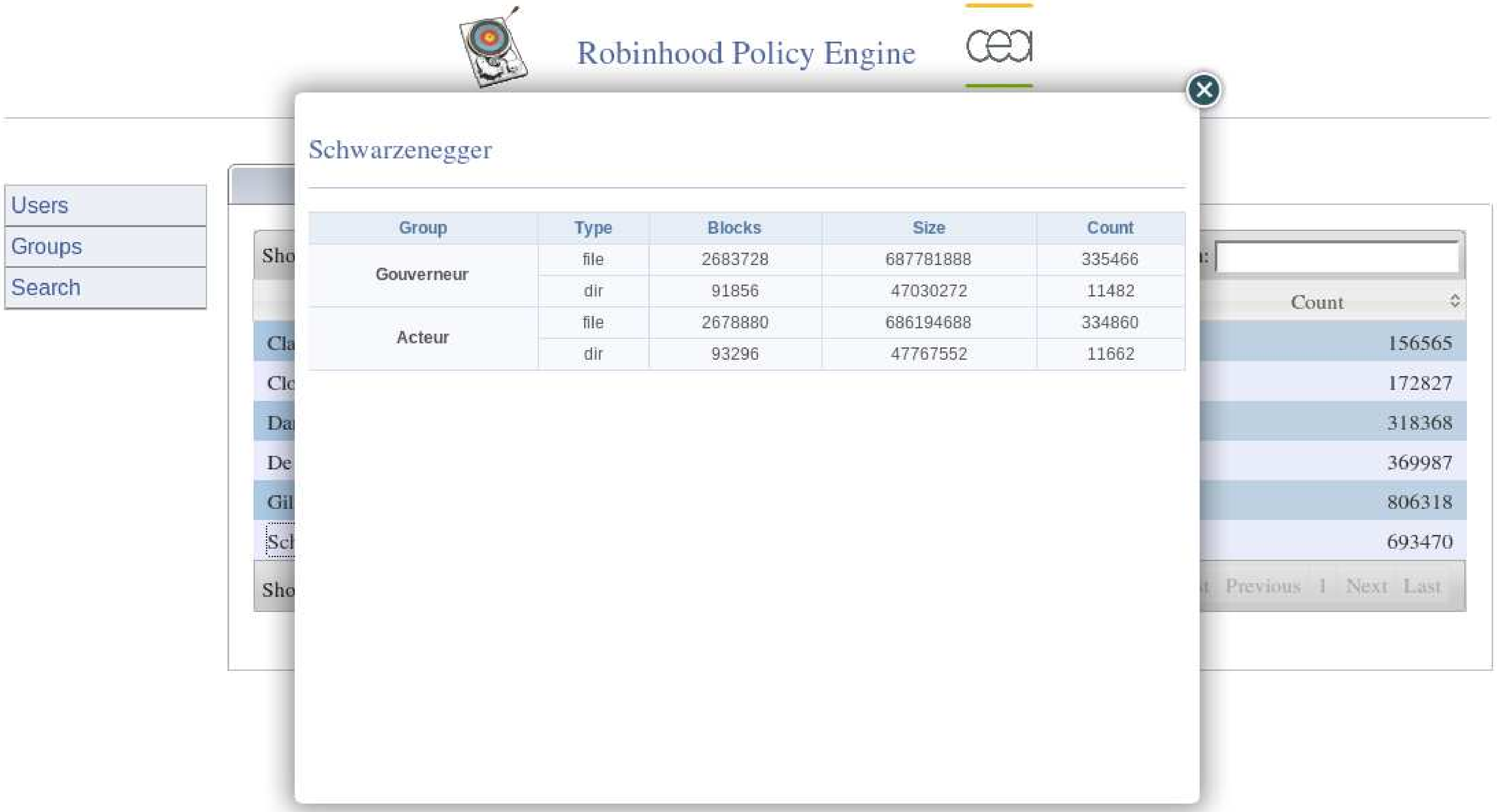
Damon: 673642496 (15.7%)
De Niro: 785629159 (18.3%)
Clapton: 502537437 (11.7%)
Others: 867668390 (20.2%)
Schwarzenegger: 1468774400 (34.2%)

User	Space used	Count
Schwarzenegger	1.37 GB	693 470
De Niro	749.23 MB	369 987
Damon	642.44 MB	318 368
Clapton	479.26 MB	156 565

View all users

Web interface

- **Detailed user stats:**



Schwarzenegger

Group	Type	Blocks	Size	Count
Gouverneur	file	2683728	687781888	335466
	dir	91856	47030272	11482
Acteur	file	2678880	686194688	334860
	dir	93296	47767552	11662

Robinhood Reporting Tool

More reports available in command-line:

- Group stats:
  - `rbh-report –g grp* --csv`

```
group,     count,        spc_used,     avg_size
grp1,      11000,     11534336000,     10485760
grp2,     101398,   3780071239680,     37286674
```

- Stats per user AND per group:
  - `rbh-report –u foo --csv -S`

```
user,      group,      count,        spc_used,     avg_size
foo,       proj1,       1542,       114336000,        74147
foo,       proj2,       1013,      3780071239,      3731560
```

- FS content summary:
  - `rbh-report –i --csv`

```
type,         count,       spc_used,     avg_size
directory,   130542,      534700032,         4096
file,       1256830,   378007123900,       300700
symlink,       1013,          30717,           30
```

- Possibly filter by directory:
  - `rbh-report –u foo –P ‘/fs/one_dir/dir*’`

Robinhood Reporting Tool

Several commands we often use:

- **Top users (by volume, inodes, avg file size):**
  - `rbh-report --topusers`
  - `rbh-report --topusers --by-count`
  - `rbh-report --topusers --by-avgsize`

Eg:

```
user,         count,          spc_used,     avg_size
foxtrot,   4912398,    3780071239680,       769496
alpha,     3423921,    1153433446000,       336875
```

- **List files per OST:**
  - `rbh-report --dump-ost 96`

- **Performance:**
  - All these commands are fast O(1) queries:

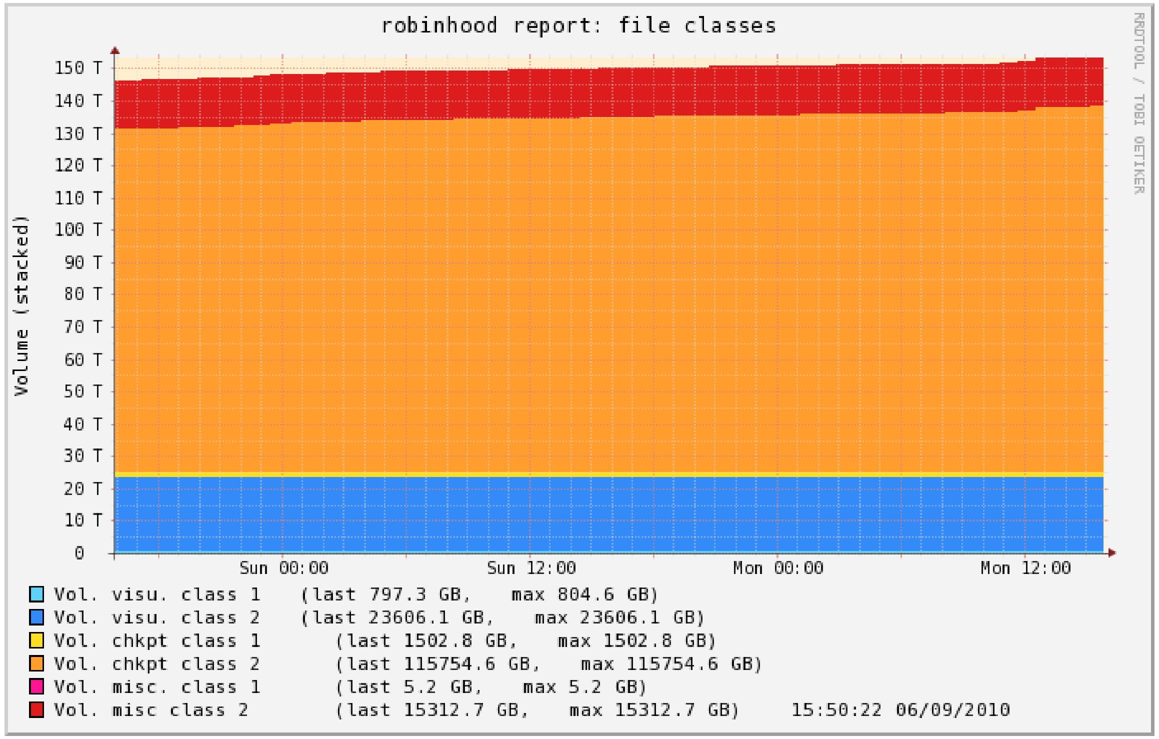
```
time rbh-report –i | -u foo | -g grp* | --topusers | …
   real    0m0.012s
   user    0m0.003s
   sys     0m0.005s
```

Profiling FS Content

- **FS content profiling using File Classes:**
  - Admin defined, based on file attributes.
    Eg:

```
FileClass SmallFiles {
   definition { type == file and size < 10MB }
}
```

  - **`rbh-report --classinfo`**



Customizable Alerts

- **Alert if anormal filesystem entries are found:**
  - Flexible, attribute-based alert definitions:

```
Alert large_file_in_bad_place {
   type == file
   and size > 1TB
   and tree != “/fs/big_files”
}
```

- **Real-life use cases:**
  - detect wide directories (>100.000k entries)
  - detect very deep namespaces (‘crazy’ code, infinite mkdir loop)
  - detect files with size == ulimit (user hit ulimit)
  - detect files in bad locations

Quota-like alerts

- **Quota-like alerts**
  - Send mail if a user/group exceeds a given threshold:

```
trigger_on = user_usage(foo*,bar*);
high_threshold_vol = 20TB;
notify = TRUE;
```

  - Based on volume or inode count

- **Real-life use cases:**
  - Filesystems that don’t have the quota feature
  - …or if the feature is unstable
  - As a complement to the quota feature (to get mail notifications)

Load Profiling

- **Changelog stats** (Lustre 2.x and robinhood >= 2.3.3)
  - Display current load profile on the filesystem:

```
 type          total     (diff)          (rate)
CREAT:      51323232  (+139225)   (2320.42/sec)
MKDIR:       4262465   (+37950)    (632.50/sec)
HLINK:          1342
SLINK:        326227     (+150)      (2.50/sec)
MKNOD:             0
UNLNK:        751223    (+1832)     (30.53/sec)
RMDIR:         23523
RNMFM:        252523
RNMTO:        252523
TRUNC:         19625
SATTR:        133232   (+13832)    (230.53/sec)
XATTR:             0
  HSM:             0
MTIME:      12532709   (+76302)   (1271.60/sec)
CTIME:             0
ATIME:             0
```

Use case 2

Managing *scratch* FileSystems

Interest of Purge Policies

- **Clean temporary files/directories**
  - Cleanup after code run (lifetime: few hours)
  - Cleanup after simulation (lifetime: several months)
  - Clean old krb tickets, logs, core dumps, crash dumps, …
  - Policies based on file properties (attributes, path, xattrs, …)

- **Limit usage per user / per group**
  - Purge trigger based on user/group usage:

    Eg:

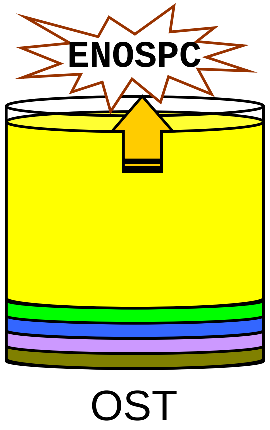
    Purge old unused files of user ‘foo’ when it exceeds 10TB or 100k inodes

- **Purge files per OST**
  - Useful before unconfiguring device or OSS

Interest of Purge Policies

- **Avoid ‘No space left on device’ errors caused by full OSTs**
  - Trigger purge if an OST exceeds a given threshold
  - Only purge files on this OST
  - Purge files by LRU until OST usage is under a given threshold
  - Files can be whitelisted, kept longer than others, …

- **Common example:**
  - Very big file with small stripe_count fills an OST
  - Example of policy to fix that:
    - Whitelist empty files
    - on OST full:
      - purge files > 500GB after 1h
      - purge other files after 12h
    - The guilty big file(s) won’t cause other recent files to be purged



Use case 3

Data archiving / HSM binding

Data Archiving

- **Robinhood schedules mass file archiving**
  - Copy modified files to external storage
  - Flow control:
    - max simultaneous copies
    - max volume / max count per *N* min window
  - Policies:
    - files can be ignored (i.e. not archived)
    - important files can be copied sooner than others
    - can specify backend-specific hints:
      - Eg. Write files > 10G to Storage Class #4
        Write files of group G1 to tape pool TP1

- **Interest:**
  - Backup filesystem data
  - Disaster recovery
  - Un-delete
  - Lustre-HSM

With and Without Lustre-HSM

- **Data archiving and HSM-binding:**
  - Lustre 2.x without Lustre-HSM (backup only):
    - Detect file modification : compare mtime/size with the archived file (not 100% safe)
    - Cannot release disk space
    - Undelete and disaster recovery
  - with Lustre-HSM (Lustre 2.?):
    - Dirty bit: Lustre built-in (safe)
    - Robinhood schedules Lustre-HSM ‘archive’, ‘release’ and ‘hsm remove’ operations
    - Automatic data recall: Lustre built-in
    - Undelete and disaster recovery

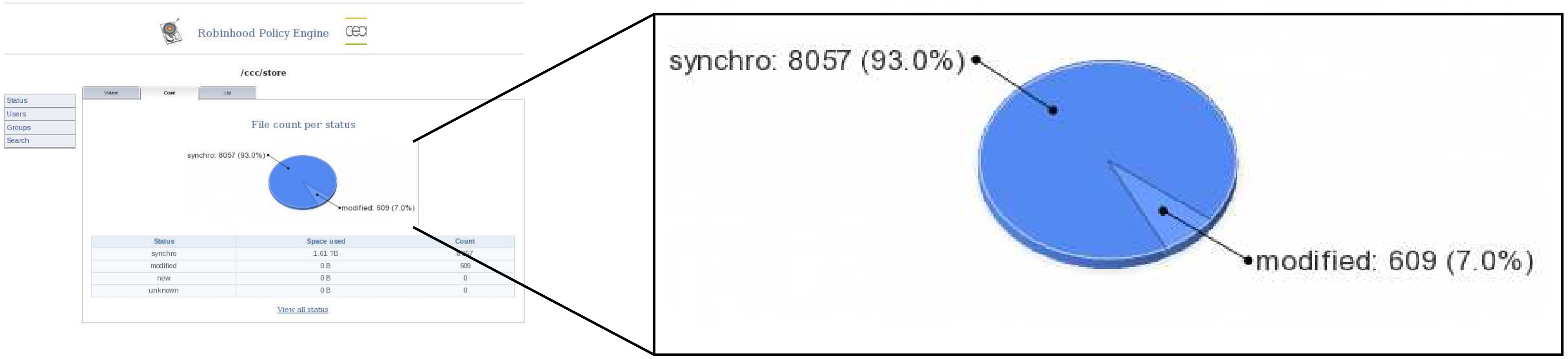
About “Undelete”

- **Robinhood keeps track of file removal:**
  - using changelog (UNLINK record)
  - if a file disappears between 2 scans

- **If an <u>archived</u> file is deleted in Lustre:**
  - Robinhood doesn’t immediately perform removal in the backend storage
  - Configurable delay for deferred removal
  - During this delay, the file can be un-deleted

Archiving Statistics

- **File status summary in the web interface:**



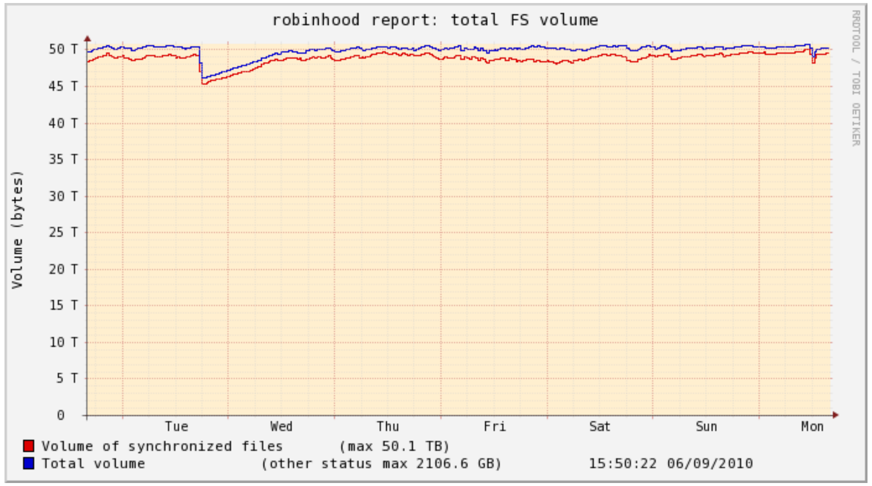
Robinhood Policy Engine

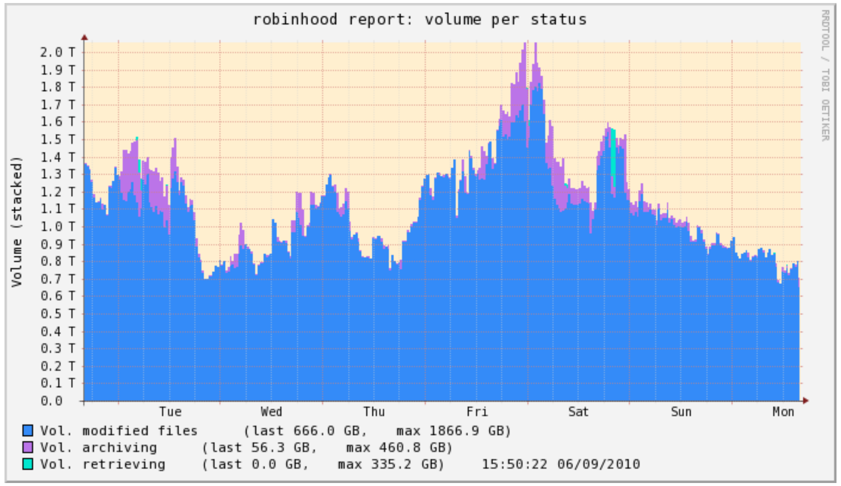
/ccc/store

Status	Space used	Count
synchro	1.61 TB	8057
modified	0 B	609
new	0 B	0
unknown	0 B	0

View all status

- **Monitoring data flows** (output of rbh-report):





Roadmap

- **Current version is 2.3.2**

Incoming features:

- **Bulk import of FS entry lists** (v2.4)
  - Import lists generated by the filesystem (e2scan, dmscanfs, …)
- ***find* and *du* clones using robinhood's DB** (*rbh-du*, *rbh-find*)
  - Faster
  - Unload the filesystem
- **Add more stats to the web interface**
- **Interface with end-users**
  - Web interface: make it possible to monitor their own usage
  - Send advices/warnings/alerts directly to file owners
- **New policies:**
  - Pool/OST migration policy
  - User-defined: use robinhood to schedule any action on FS entries
- **Take a look at NOSQL databases**
  - Scale over billions of entries
  - Process higher FS event rates

Wrap up

- **Keep an eye on your filesystem content to ensure a good Quality of Service**
  - Identify main resource consumers
  - Detect ‘crazy’ codes
  - Watch harmful behaviors
  - Storage resource sizing

- **Unload your filesystem / optimize your searches**
  - Replace *find*-based scripts by SQL queries to robinhood DB

- **Apply fair and flexible purge policies efficiently**

- **Save your filesystem data:**
  - data archiving and ‘undelete’

➢ **Get started here: http://robinhood.sf.net**

RobinHood Policy Engine

`http://robinhood.sf.net`

European Lustre Workshop 2011

September 26-27, 2011

Thomas LEIBOVICI
thomas.leibovici@cea.fr