FROM RESEARCH TO INDUSTRY

www.cea.fr

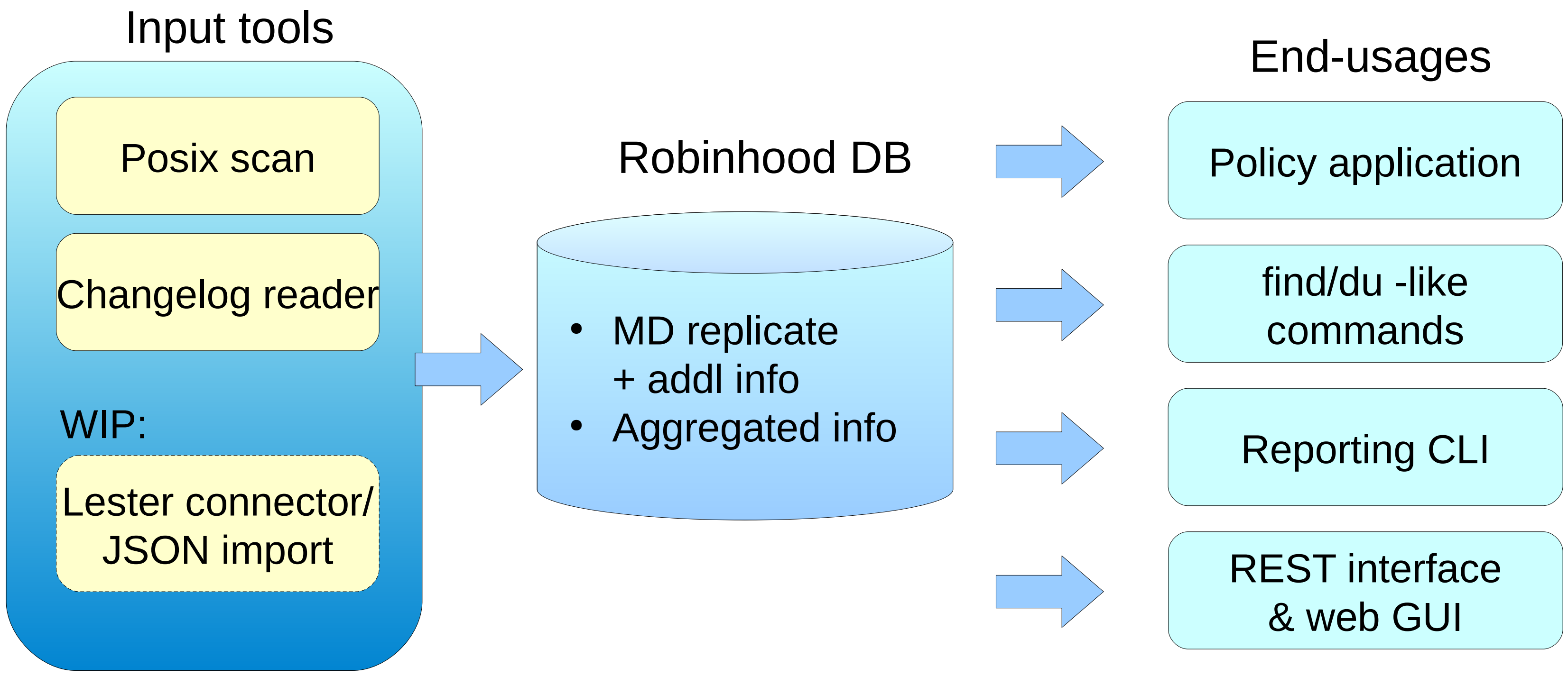
Robinhood status and latest improvements

LAD'18

Thomas Leibovici <thomas.leibovici@cea.fr>

September 25th, 2018

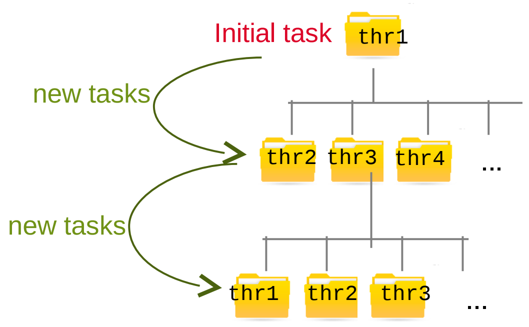
Robinhood Big Picture



POSIX SCAN

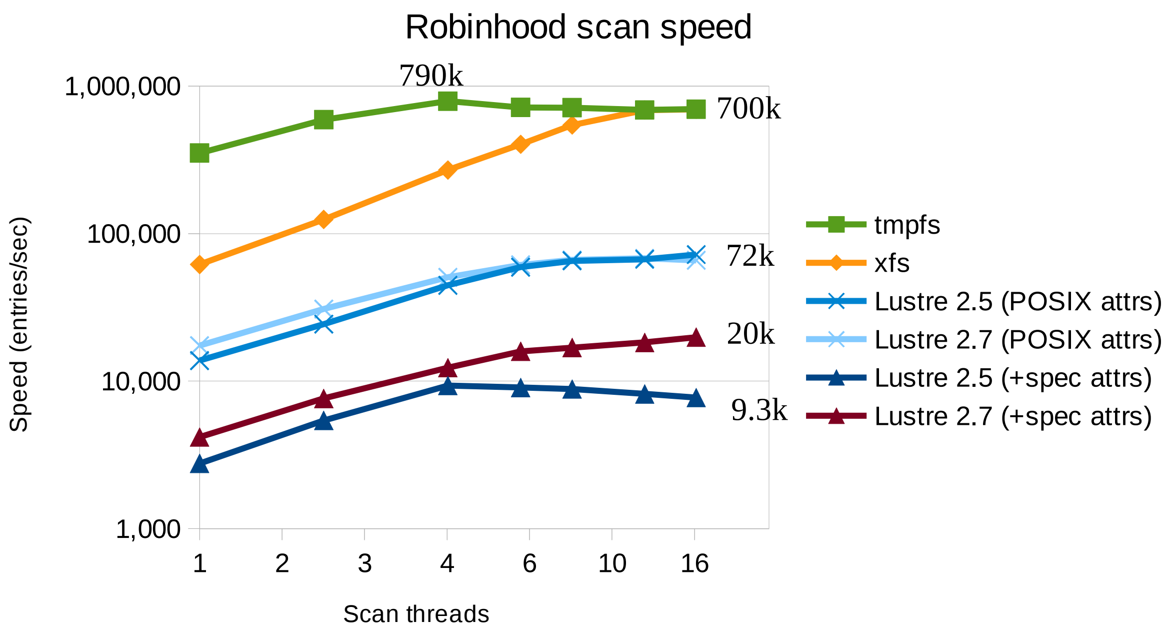
Principle

- Multi-threaded scan



Performance

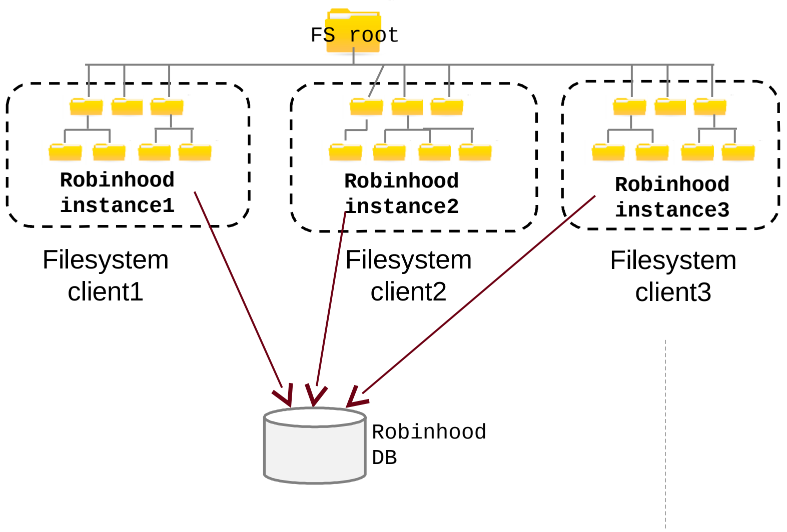
- ~800k entries/sec on a local FS
- 10 to 80 times slower with Lustre



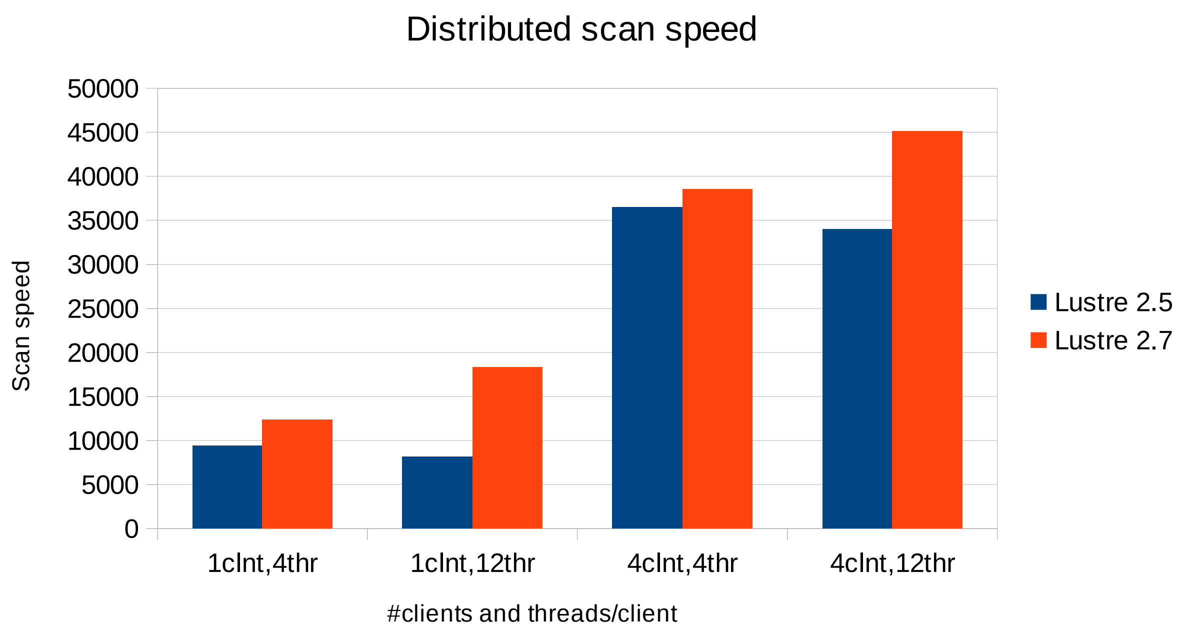
PARALLEL SCAN (MULTI-CLIENT)

Principle

- Scan can be distributed across multiple clients
  - Split the namespace between multiple scanners
- Allows cumulating ops/sec of individual clients



Performance



ex. configuration :

```
scan_only = /dir1;
scan_only = /dir2;
...
```

Changelog reader

Principle

- Read MD updates from Lustre changelogs (CREATE, MKDIR, CLOSE, HSM...) apply to Robinhood DB near real-time
- Changelogs are stored until Robinhood acknowledges them (no event is lost)

Details

- 1 reader per MDT
- Possible setup:
  - all readers in a single process
  - multiple processes on the same host
  - distributed on multiple hosts

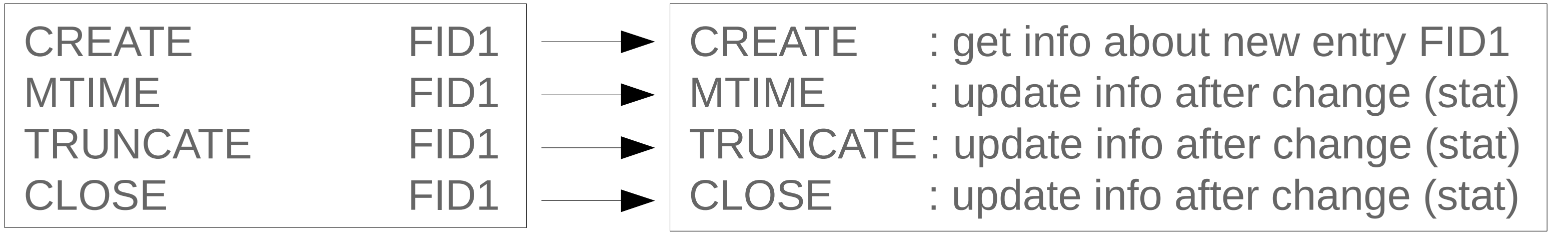
```
robinhood --readlog
```

```
robinhood --readlog=<mdt_index>
```

Changelog aggregation

Optimization: changelog aggregation

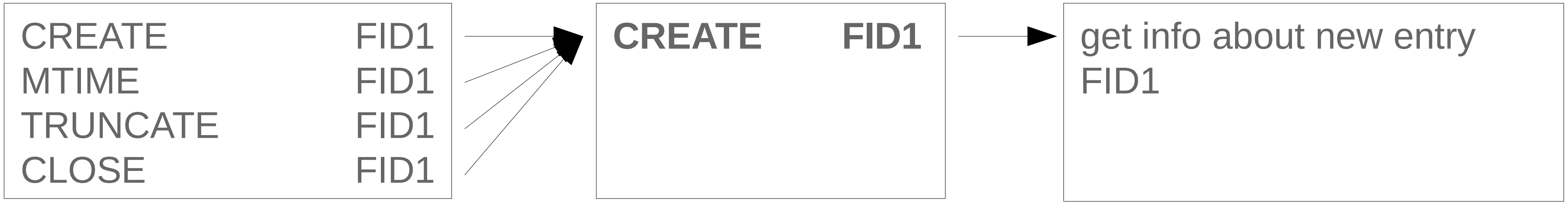
- Without optimization:



CREATE FID1
MTIME FID1
TRUNCATE FID1
CLOSE FID1
CREATE : get info about new entry FID1
MTIME : update info after change (stat)
TRUNCATE : update info after change (stat)
CLOSE : update info after change (stat)

Incoming records Robinhood processing

- Changelog aggregation (since robinhood 2.5): batch processing of changelogs with similar processing



Incoming records

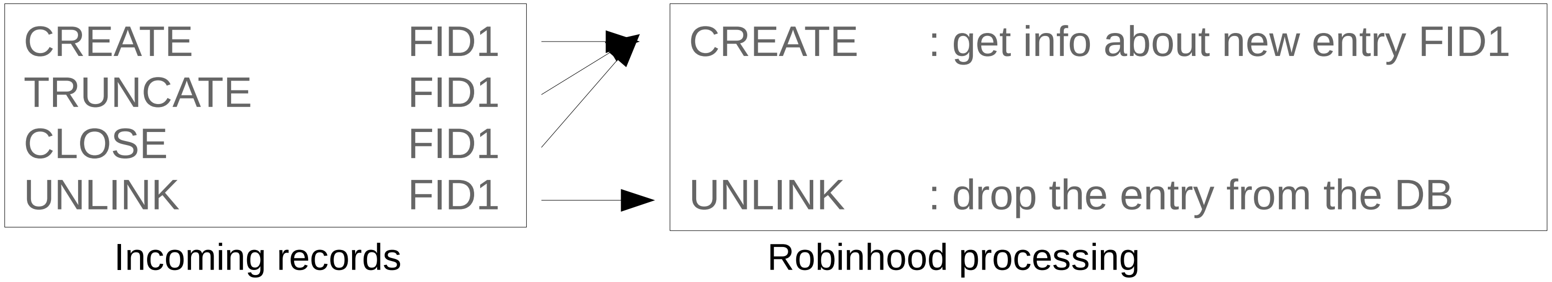
Changelog reader dedup queue

Robinhood processing

Enhanced Changelog aggregation

Enhanced changelog aggregation

- With previous optimization:



CREATE	FID1
TRUNCATE	FID1
CLOSE	FID1
UNLINK	FID1

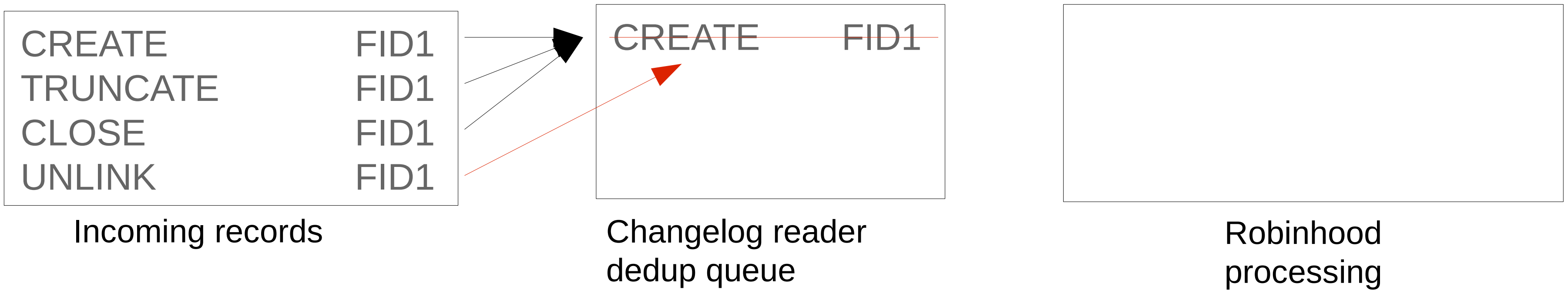
Incoming records

CREATE : get info about new entry FID1

UNLINK : drop the entry from the DB

Robinhood processing

- Enhanced changelog aggregation (since robinhood 3.1.4):



CREATE	FID1
TRUNCATE	FID1
CLOSE	FID1
UNLINK	FID1

Incoming records

~~CREATE FID1~~

Changelog reader dedup queue

Robinhood processing

Changelog performance

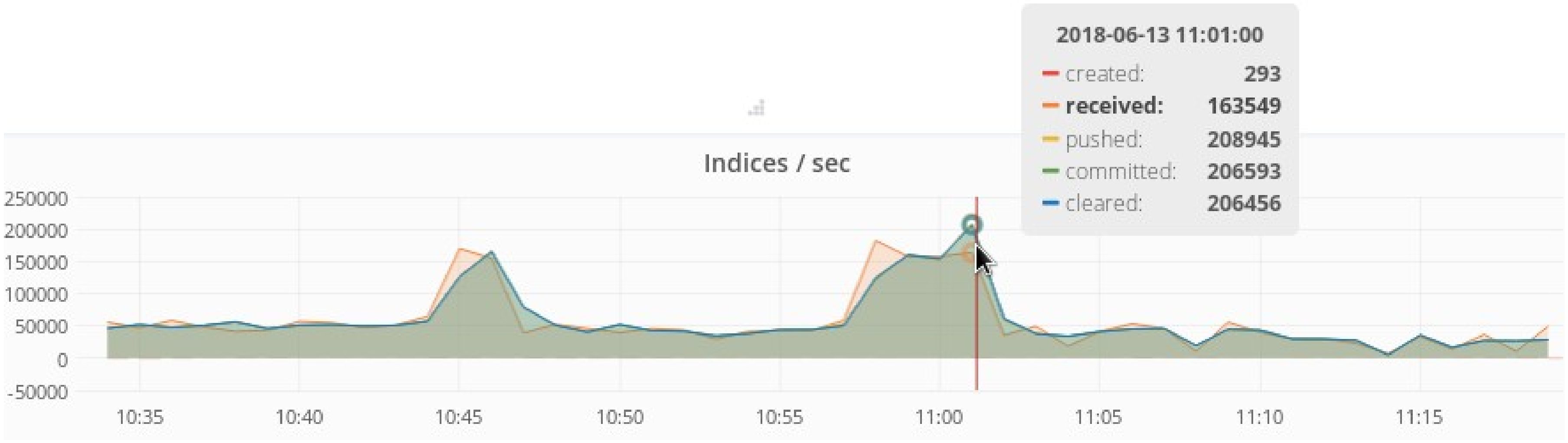
Resulting performance

- Example on a scratch filesystem:

```
records read          = 8889464866
interesting records = 1533615524
suppressed records  = 7159476712
```

-80% records

- Performance observed: up to 200k records/sec



2018-06-13 11:01:00

created:	293
received:	**163549**
pushed:	208945
committed:	206593
cleared:	206456

Changelog performance tips

Performance tips

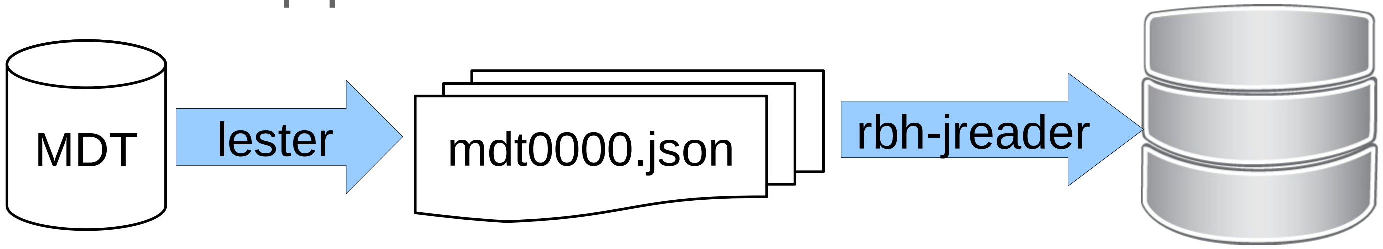
- Get the latest version of robinhood (3.1.4)
- Increase changelog aggregation window depending on host memory:
  - Parameters “queue_max_size” and “queue_max_age”
- Run changelog reader in a separate process (dedicated memory allocator)
- Run mariadb with jemalloc:
  - In `/etc/systemd/system/mariadb.service.d/override.conf`:

```
[Service]
Environment="LD_PRELOAD=/usr/lib64/libjemalloc.so.1"
```

(WIP) Lester connector / rbh-jreader

Work in progress: lester connector for robinhood

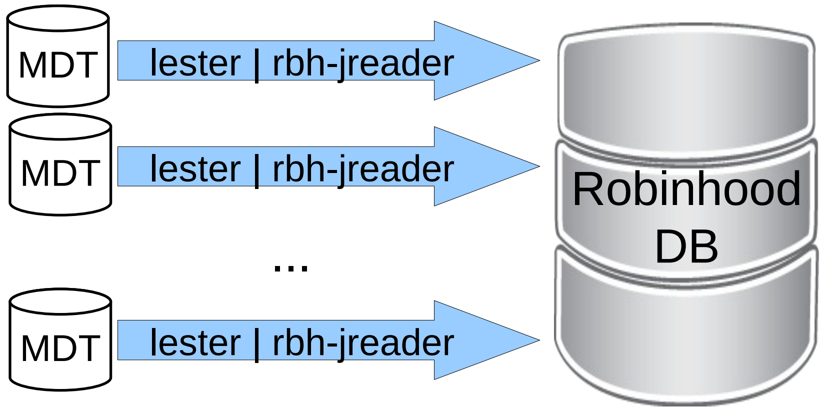
- Lester: tool developed by ORNL
  - Directly scans the MDT inode table
- Lester contributions:
  - JSON output
  - dump extra information: link EA, hsm EA
- New robinhood command: rbh-jreader
  - Read JSON output from lester and inject it to robinhood DB
  - File or pipe



(WIP) Lester connector: benefits and performances

Benefits and performances

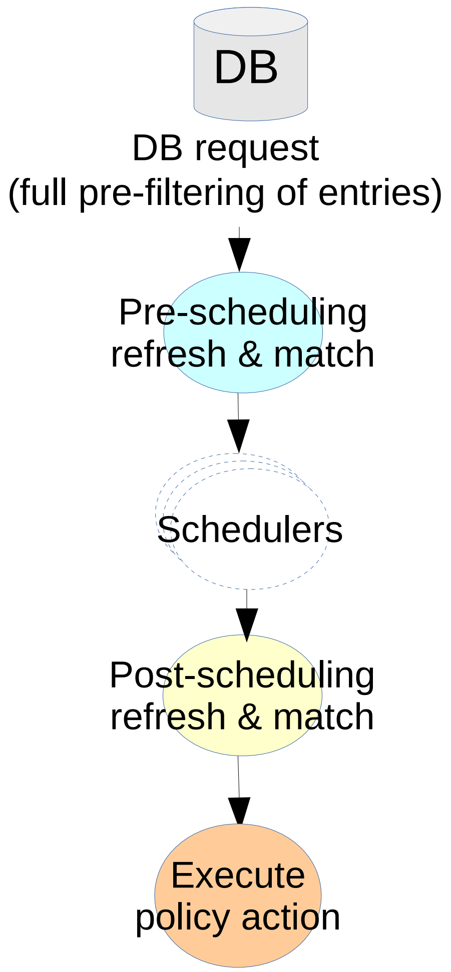
- Low level ldiskfs scan: fast!
- Takes advantage of distributed metadata



Scan duration	
Lester + rbh-jreader	Namespace scan
3 h 22 min	33 h 8 min

Distribution of inodes across MDTs									
0	1	2	3	4	5	6	7	8	9
126 540	**144 790 552**	125 318	125 317	1 087 023	141 998	141 998	125 306	301	8 958 952

Policy run enhancements



Enhancements (robinhood 3.1):

- Full conversion of policy rules to DB request
→ minimizes entries to be processed

- Smarter and configurable matching behavior (before and after scheduling)

- Schedulers :
  - Can delay, reorder, skip entries...
  - Stackable
  - Plugins (you can implement your own)
  - Provided implementation: “common.rate_limit”
    - Allow limiting the rate of actions (count and/or size)
    - Especially useful for Lustre/HSM archive

Policy schedulers: example

```
lhsm_archive_parameters {

    # limit archive rate to avoid flooding the MDT coordinator
    schedulers = common.rate_limit;

    rate_limit {
        # max count per period
        max_count = 10000;
        # max size per period: 10GB/s
        max_size = 100GB;
        # period, in milliseconds: 10s
        period_ms = 10000;
    }

...
```

REST interface

REST API

- Allow querying robinhood DB using a simple HTTP client
- Fine-grained access control (per user, per group, IP, hostname...)
- Package “robinhood-webgui”

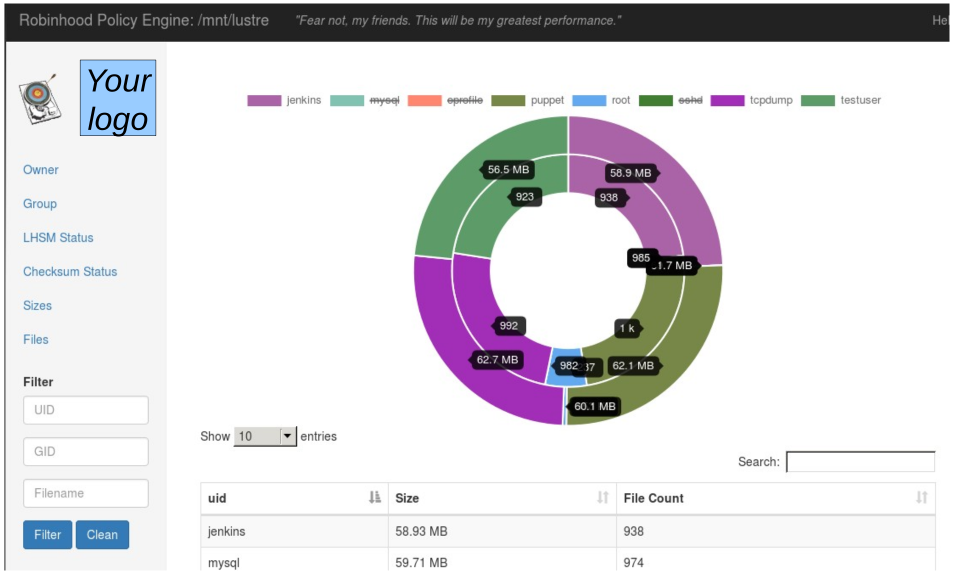
- Request format:
  `<server>/api/native/{acct|data|graph}/`**`field.operator/field.operator/...`**
- e.g. sum all information per group:
  `https://rbh-host/api/native/acct/`**`gid.group/size.desc`**

```
[{
       "gid": "somegroup",
       "size": "14406475930640792",
       "count": "6813043",
            …
},
...
```

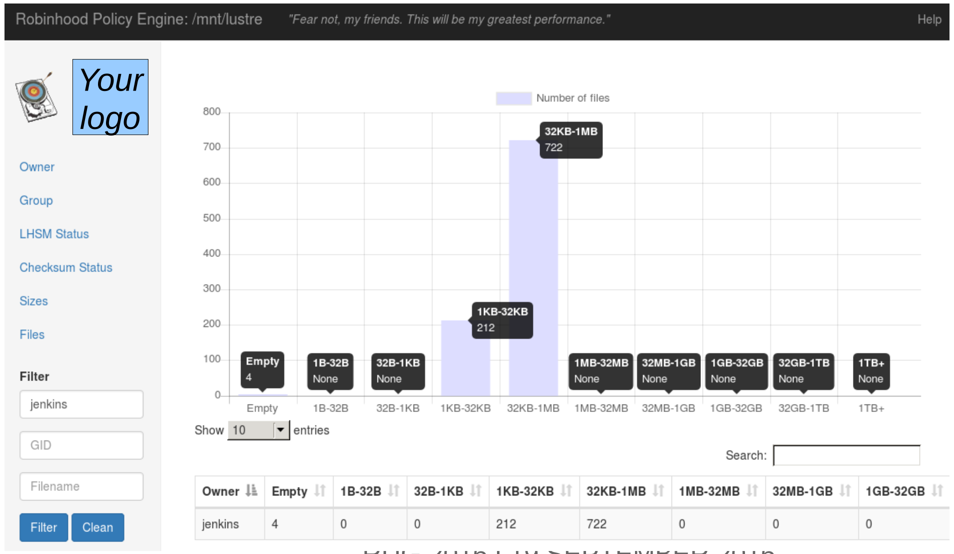
- Full API documentation: `/var/www/robinhood/README.txt`

Web interface

- New web gui in v3
- File distribution per user, group, size, policy status...
- Filtering
- Namespace browsing
- New features in 3.1.4:
  - Tasks
  - Custom graphs



uid	Size	File Count
jenkins	58.93 MB	938
mysql	59.71 MB	974

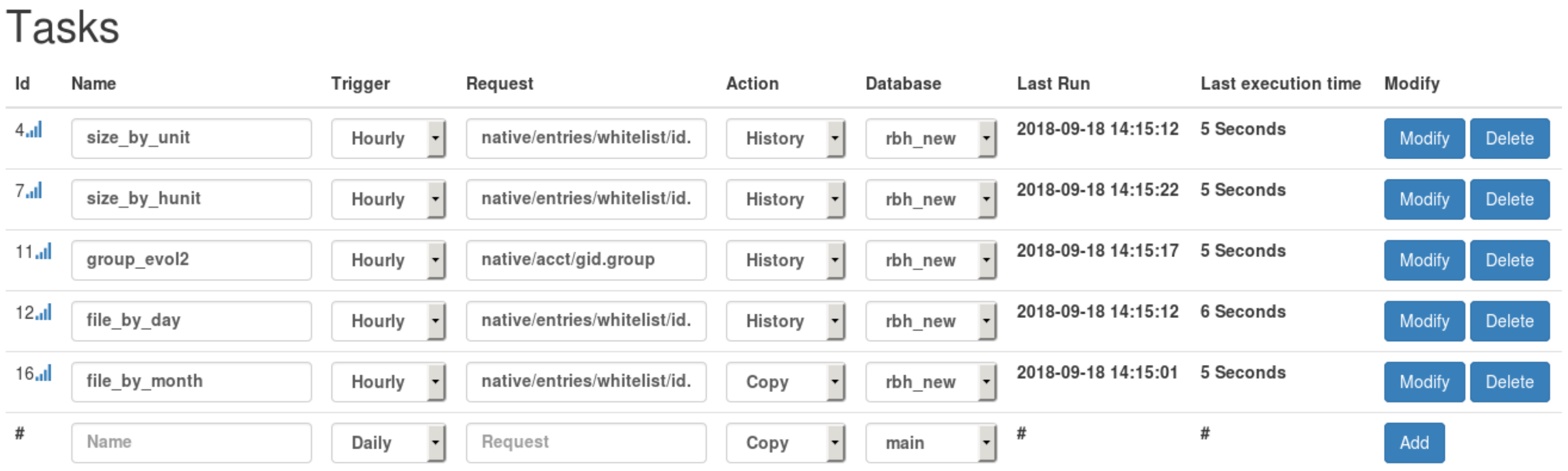


Owner	Empty	1B-32B	32B-1KB	1KB-32KB	32KB-1MB	1MB-32MB	32MB-1GB	1GB-32GB
jenkins	4	0	0	212	722	0	0	0

Web interface: tasks

Scheduled requests (tasks)

- Run custom REST requests at scheduled interval
- Possibility to keep an history of request results, to graph stats evolution
- Target database selection
- 1 click to turn it to a custom graph



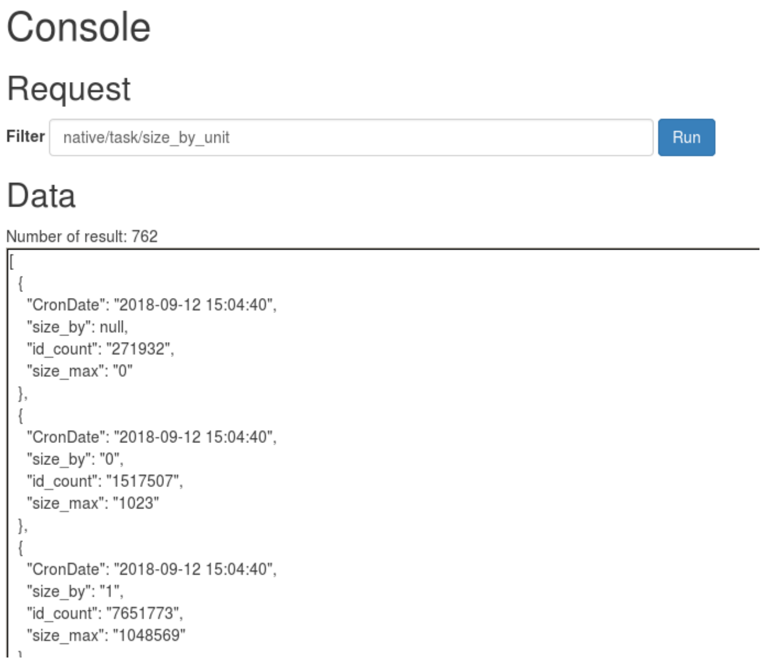
Tasks

Id	Name	Trigger	Request	Action	Database	Last Run	Last execution time	Modify
4	size_by_unit	Hourly	native/entries/whitelist/id.	History	rbh_new	2018-09-18 14:15:12	5 Seconds	Modify Delete
7	size_by_hunit	Hourly	native/entries/whitelist/id.	History	rbh_new	2018-09-18 14:15:22	5 Seconds	Modify Delete
11	group_evol2	Hourly	native/acct/gid.group	History	rbh_new	2018-09-18 14:15:17	5 Seconds	Modify Delete
12	file_by_day	Hourly	native/entries/whitelist/id.	History	rbh_new	2018-09-18 14:15:12	6 Seconds	Modify Delete
16	file_by_month	Hourly	native/entries/whitelist/id.	Copy	rbh_new	2018-09-18 14:15:01	5 Seconds	Modify Delete
#	Name	Daily	Request	Copy	main	#	#	Add

Web interface: custom graphs

Building custom charts

- 1) In console panel, test your request and see what your graph looks like
- 2) Give it a name and click "Add custom graph"



Console

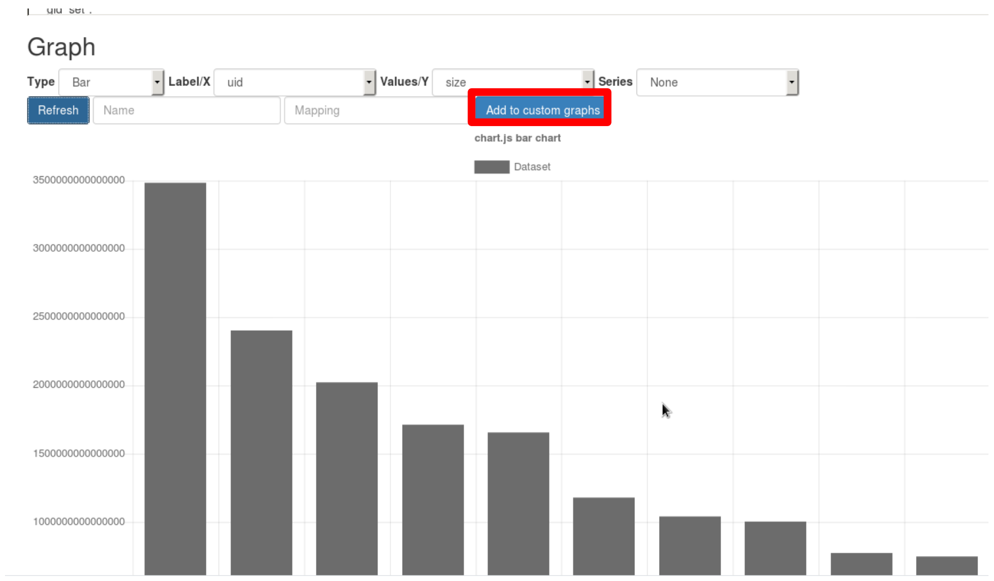
Request

Filter: native/task/size_by_unit

Data

Number of result: 762

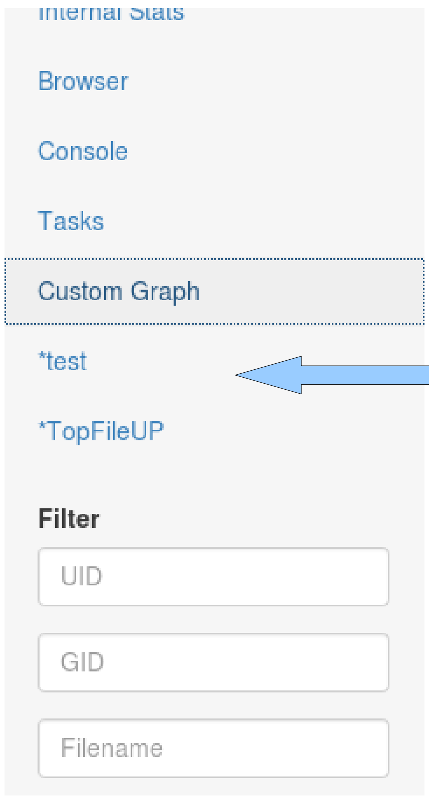
```
[
 {
  "CronDate": "2018-09-12 15:04:40",
  "size_by": null,
  "id_count": "271932",
  "size_max": "0"
 },
 {
  "CronDate": "2018-09-12 15:04:40",
  "size_by": "0",
  "id_count": "1517507",
  "size_max": "1023"
 },
 {
  "CronDate": "2018-09-12 15:04:40",
  "size_by": "1",
  "id_count": "7651773",
  "size_max": "1048569"
```



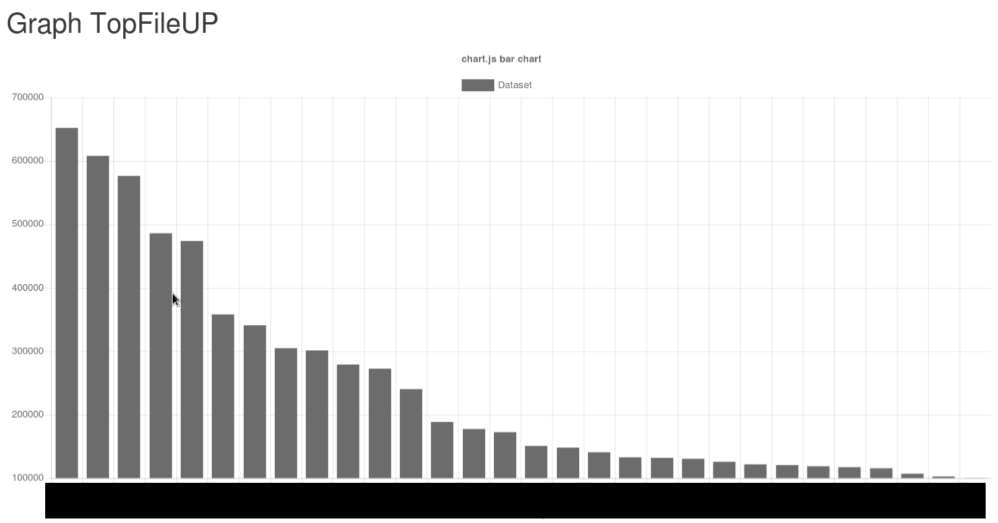
Graph

Type: Bar · Label/X: uid · Values/Y: size · Series: None

Web interface: custom graphs



Custom graphs appear as new items in the menu



[redacted]

Thank you for your attention!

Questions?

Commissariat à l’énergie atomique et aux énergies alternatives
CEA / DAM Ile-de-France | Bruyères-le-Châtel - 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00

DAM Île-de-France

Policy application

Defining a policy

- 1) Use a template policy, or define a custom policy
  - E.g. lhsm_archive, cleanup, rmdir...
- 2) Define file classes (based on entry attributes)
  - E.g. `name == *.log and (size > 1GB or owner == foo)`
- 3) Define policy rules indicating:
  - Target file classes
  - “last minute” criteria (e.g. time-based)
  - Action and parameters
- Define policy triggers if you plan to run the policy in robinhood daemon
- 4) Run the policy
  - On all matching entries in the filesystem
  - Or, limited to a user, group, file class, OST set...

Example: pool migration

Example: migrating files to specific pools

- 1) Policy definition (custom)

```
define_policy move2pool {
    status_manager = basic;
    scope { type == file and status != ok }
    default_action = cmd("migrate2pool.sh {tgt_pool} {path}");
}
```

- 2) Define target fileclasses

```
fileclass wrong_pool_big {
    definition { size > 100GB and ost_pool != "big_pool" }
    move2pool_parameters { tgt_pool = big_pool; }
 }
}
fileclass wrong_pool_small {
    definition { size < 10MB and ost_pool != "small_pool" }
    move2pool_parameters { tgt_pool = small_pool; }
 }
}
```

Example: pool migration (cont'd)

- 3) Policy rules:

```
move2pool_rules {
    rule rulexxx {
         target_fileclass = wrong_pool_small;
         target_fileclass = wrong_pool_big;
         condition {last_mod > 6h}
    }
}
```

- 4) Running the policy: command examples

```
# run regularly on all matching entries (daemon)
robinhood --run=move2pool -d

# run once on user foo
robinhood --run=move2pool --target=user:foo

# run once on OST 42
robinhood --run=move2pool --target=ost:42
```

Reporting commands

- **rbh-report**: query robinhood DB (entry info, reports, ...)
- **rbh-find**: “find”-like command to search for entries according to various criteria (attributes, striping, policy status, …)
- **rbh-du**: “du”-like command with additional filtering features
- **rbh-diff**: display differences between current FS state and rbh DB.