Take back control with RobinHood v3

LUG'17

Henri Doreau <henri.doreau@cea.fr>

June 1st 2017
Project History

Robinhood Policy Engine

- **Mature**
  - Development started in 2005
  - Constantly improved since then
  - Now widely used in HPC centers of various size
  - Large contributors base (sites, vendors...)

- **Open Source**
  - Initially developed for internal needs
  - Open sourced in Feb. 2009 (now lives on http://github.com/cea-hpc/robinhood)

- **Versatile**
  - Purging entries on temporary filesystems
  - Conductor of Lustre/HSM installations
  - Rich reporting and near-real time monitoring
  - Powerful suite of companion tools
# Robinhood 3 in a Nutshell

<table>
<thead>
<tr>
<th>v2 “flavors” and their commands</th>
<th>A static set of available policies per flavor</th>
<th>V3: a single instance to manage all “legacy” policies and much more!</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>robinhood-tmpfs</strong></td>
<td><strong>robinhood-lhsm</strong></td>
<td><strong>robinood-backup</strong></td>
</tr>
<tr>
<td>robinhood</td>
<td>rbh-lhsm</td>
<td>rbh-backup</td>
</tr>
<tr>
<td>rbh-diff</td>
<td>rbh-lhsm-diff</td>
<td>rbh-backup-diff</td>
</tr>
<tr>
<td>rbh-report</td>
<td>rbh-lhsm-report</td>
<td>rbh-backup-report</td>
</tr>
<tr>
<td>rbh-du</td>
<td>rbh-lhsm-du</td>
<td>rbh-backup-du</td>
</tr>
<tr>
<td>rbh-find</td>
<td>rbh-lhsm-find</td>
<td>rbh-backup-find</td>
</tr>
</tbody>
</table>

→ Policies declared in configuration
Robinhood Policy Engine: overview

- Collects information about filesystems
  - Maintain a up-to-date image of filesystem metadata
  - Lustre: based on MDT changelogs
  - Posix: periodic scanning

- Define custom policies to schedule actions on filesystems entries
  - v2.x: archiving data, purging scratch filesystems, HSM...
  - v3+: way much more!
  - Flexible, fine-grained policy rules

- Provides an overall view of filesystems contents
  - File size profile per user, per group, …
  - Classifying entries in arbitrary admin-defined sets (fileclasses)

- A set of convenient utilities to manage Lustre filesystem contents efficiently
  - rbh-find, rbh-du, rbh-diff...
Robinhood Policy Engine

Big picture

Parallel scan (once)

Lustre v2 ChangeLogs

near real-time DB update

Robinhood database

Admin rules & policies

find and du clones

Fine-grained statistics + web UI

Mass action scheduling (policies)

Attribute-based alerts

Disaster recovery helpers

Generic Lustre/HSM copytool
Filesystems and Databases

Respective benefits

**Data intensive workloads**

**Filesystem**

- **Goals**
  - Optimize data access
    - Bandwidth, data allocation
  - Optimize medatada access for POSIX
    - lookup/readdir/create/unlink

  lfs find . -user foo -size $1024 | wc -l

**Database**

- **Goals**
  - Optimize per-record access
    - select/insert/update
  - Optimize multi-criteria searches
  - Optimize aggregating/sorting information

  \[ \text{select count(*) from ENTRIES where user=‘foo’ and size<1024} \]
Robinhood v3 Plugin-Based Architecture

- **Purpose**
  - Purpose-specific code moved out of robinhood core: now dynamic plugins loaded at run-time
  - All policy behaviors made configurable
  - Vendors/users can write their own plugins for specific needs

- **Core**
  - Entry processing, policy scheduling, rules matching, reporting...

- **Managed Filesystem**

- **Status manager plugins**
  - Lustre/HSM
  - Data migration
  - Custom...

- **Action plugins**
  - Common (unlink, copy...)
Generic Policies (v3.0): Motivation

Before v3

- Static set of policies, statically defined
- 1 mode = 1 robinhood instance = 1 set of commands
- Instances can't coexist on the same filesystem

<table>
<thead>
<tr>
<th>Package</th>
<th>&quot;migration&quot; policy</th>
<th>&quot;purge&quot; policy</th>
<th>&quot;hsm_remove&quot; policy</th>
<th>&quot;rmdir&quot; policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>robinhood-tmpfs</td>
<td></td>
<td>rm (old files)</td>
<td></td>
<td>rmdir, rm –rf</td>
</tr>
<tr>
<td>lustre/posix</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>robinhood-backup</td>
<td>Copy to storage</td>
<td>-</td>
<td>rm in storage</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>backend</td>
<td></td>
<td>backend</td>
<td></td>
</tr>
<tr>
<td>robinhood-lhsm</td>
<td>Lustre HSM archive</td>
<td>Lustre HSM release</td>
<td>Lustre HSM remove</td>
<td></td>
</tr>
</tbody>
</table>

Robinhood v2.x packages and policies

- E.g. Lustre/HSM purpose
  - Package: robinhood-lhsm
  - Commands: rbh-lhsm-*
  - Only implements HSM-related policies (archive, release, remove)
  - Cannot manage other actions (delete old files, …)
Robinhood v3

- A single Robinhood instance for all purposes:

<table>
<thead>
<tr>
<th>Package</th>
<th>Generic policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>robinhood-lustre</td>
<td>Fully configurable</td>
</tr>
</tbody>
</table>

- Lustre filesystems:

- Other filesystems:

- Robinhood core: **generic** policy implementation

- Specific aspects:
  - Specified by **configuration** (policy templates)
  - Possibly as specific **plugins** (dynamic libraries)

- Policies at will
  - Schedule any conceivable action
  - Just by writing a few lines of configuration
Example: configurable pool migration with just a few lines of config

- Declare policy

```plaintext
declare_policy move_pool {
    scope { type == file and status != ok }
    default_action = cmd("lfs migrate -p {pool} -c {count} {path}";
    status_manager = basic ; # manages ok/failed status
}
```

- Specify rules

```plaintext
move_pool_rules {
    rule migr_movies {
        target_fileclass = movie_types;
        action_params { pool = "pool1"; count = 2; }
        condition { last_mod > 6h }
    }
    rule migr_hpc_data {
        target_fileclass = big_hpc_files;
        action_params { pool = "pool2"; count = 16; }
        condition { last_mod > 6h }
    }
}
```
Rbh-report: see what is going on

Examples of reports

- **Inode count and volume usage**
  
  ```
  $ rbh-report -u foo* -S
  user, group, type, count, spc_used, avg_size
  fool, proj001, file, 422367, 71.01 GB, 335.54 KB
  ...
  Total: 498230 entries, 77918785024 bytes used (72.57 GB) 00
  ```

- **File size profiles per user, per group**
  
  ```
  $ rbh-report --szprof -i|u 'foo*'|g 'bar*'  
  ```

- **Printf option to rbh-find (contributed by Cray)**
  
  ```
  $ rbh-find -status lhsm:released -printf "%p %Rm{lhsm.archive_id}\n"
  ```

- **Top users, top groups, top file sizes, top directories**

- **Changelog statistics: operations rate (create, mkdir, setattr...)**
Nice new features since last year

New web interface (in 3.0)

- New WebUI, compatible with robinhood 3 DB schema
- Modern widgets and layout
- Fine-grained authentication
- Compatibility with newer MySQL versions
Nice new features since last year

**REST interface (in 3.0)**

- Makes it possible to query robinhood DB through a standard protocol (HTTP)
- 3 possible output format:
  - Classic JSON (key-value): http://server/api/native/...
  - Datatables.js: http://server/api/data/...
  - GraphJS: http://server/api/graph/...

- Simple and convenient query language:
  - Returns usage stats about all users and status (as JSON)
    http://rbh/api/native/acct/...
  - Returns usage stats about a given user (as JSON)
    http://rbh/api/native/acct/uid.filter/foo

Advanced querying. Example: split user's info by gid
http://rbh/api/native/acct/uid.filter/foo/gid.group

- Allow querying robinhood stats from scripts, dashboards, …
  - E.g: take usage stats into account for job scheduling
Plugins: extending robinhood
Example of plugin: “checker” (v3.0)

“Checker” policy plugin

- Executes admin-defined commands and stores their output to rbh's DB
- Saves OK/failed status
- Manages specific attributes: last execution time and last success time
- Example applications:
  - **Detecting silent corruption**: run “md5” on files at regular interval, and check the output is unchanged.
  - **Audit filesystem contents**: run “file” utility on all files, then generate a report by file type
    
    ```sql
    SELECT ... GROUP BY file_output
    ```
Example of plugin: “modeguard”

Community-contributed policy plugin

- Enforces mode on selected entries
- Maintains OK/Invalid status on entries
- Two parameters: “set mask” and “clear mask”

Example applications:

- **Force user directories to be setgid**: set_mask=02000
- **Remove executable bits on files**: clear_mask=0111

Again: the scope of the policy is defined in the configuration
Developers: how to write a policy plugin? (1/2)

Anatomy of a robinhood plugin

- Plugins are Shared Object Libraries
  - Loaded on demand
  - Cached by the application
  - Can be included within the project or distributed separately

- Expose a clearly defined interface
  - `mod_get_name()`
  - `mod_get_version()`
  - `mod_get_{status_manager, action, scheduler}()`
Exposed methods (details)

- Pick a name
- Define the parameters of your module
- Define the status manager
  - Set of all possible states of an entry
  - How to store them in the DB (type, default value...)
  - A couple callbacks for rbh to operate the state machine
- Define the exposed actions
  - Core functions of the policy
  - Set mode, rename file, delete directory, archive file...
- See the existing ones in: [http://github.com/cea-hpc/robinhood/src/modules](http://github.com/cea-hpc/robinhood/src/modules)
Development status and roadmap
New in v3.1: schedulers

Problem: how to regulate the pace of actions and order them properly?

- 1\textsuperscript{st} example: avoid overwhelming the coordinator with archive requests
  - No existing feedback mechanism from MDT to Robinhood

- 2\textsuperscript{nd} example: archive into a rate-limited system
  - Interleave big and small files to maximise rate and throughput
Schedulers

Implemented as plugins

- Enabled and parametrized from configuration files
- Stackable
- Entry handling function can decide to:
  - Take the entry (forward it to the next level of processing)
  - Skip the entry for this run
  - Pause the handling of new entries for a while
  - Stop the handling of new entries for this run
  - Stop and cancel in-flight entries in the other schedulers
Upcoming features

Robinhood v3.1 (1H2017)

- Fixes from 3.0
- Schedulers
  - TBF rate limiting
  - Per run-limitations
- New policy plugins
  - Modeguard
  - Deferred purges
- Performance improvements
- Improved GUI
Robinhood v3 Roadmap

Candidate features for v3.2 (2H2017)

- **Asynchronous 'stat' of entries**: higher ingest rate
  - No 'stat' performed synchronously when processing changelogs
  - Changelog are ingested directly to the DB (high throughput!)
  - Background (asynchronous) update of entry metadata in DB

- **Asynchronous accounting**: more information, reduced impact on performance
  - Reduce the impact of 'accounting' on DB performance
    - Can possibly be offloaded to a 2nd server
  - Allows implementing much more aggregated stats (track users activity, jobs activity...)

Next Plans: Asynchronous Accounting

Asynchronous accounting

- Goal: reduce the impact of accounting on ingest rate.
- Make it possible to distribute the accounting processing and its DB.

Current DB workflow:

1. Incoming information
2. Synchronous update
3. Aggregated stats per user, group, type, status...

Single DB engine

Asynchronous accounting:

1. Incoming information
2. Increment queue (lockless)
3. Dequeue (async)
4. Acct updater

Main DB engine

Annex DB engines

- User accounting
- Job accounting
- Directory accounting
Lustre contributions from the robinhood project

Misc. performance and stability enhancements

- **New changelog distribution interface**
  - Character device to efficiently deliver records from kernel to userland
  - Orders of magnitude faster than the venerable “KUC” pipe
  - Landed for 2.10 (LU-7659)

- **QoS for HSM requests on the coordinator**
  - Reduce the impact of massive archiving campaigns on Lustre/HSM
  - Target 2.10 (LU-9482)

- **New LustreAPI**
  - Work by Cray tracked by LU-5969
  - Optimize massive entry handling
    - Avoid continuous open/close of FS root and “fid” directory for IOCTLs
Getting involved

What can robinhood do for you?

- **Administrators**
  - Install (or upgrade to) v3
  - Give us feedback on the mailing lists ([robinhood-support@sf.net](mailto:robinhood-support@sf.net))
  - Tell us about the limitations you encounter, the features you would need

- **Developers**
  - Implement new plugins and make people happy
  - Help experimenting with alternative DBMS
  - Get in touch on [robinhood-devel@sf.net](mailto:robinhood-devel@sf.net)

- **Vendors**
  - Consider the added value of solution-specific plugins
Thank you for your attention!

Questions?