FROM RESEARCH TO INDUSTRY

RobinHood v3

Robinhood User Group 2015

Thomas Leibovici <thomas.leibovici@cea.fr>

SEPTEMBER, 21st 2015

www.cea.fr

About Robinhood v3

Next major release: robinhood v3.0

- Design started early 2014
- In development since mid-2014
- Cray joined v3.0 development early 2015

What is it about?

- Use the policy engine to implement an infinity of use cases
  - e.g data migration between OSTs, trash mechanisms, integrity checks, data conversions...
  - Allow many new usages
- Address previous lacks, improve unhandy features
- Robustness & code quality
- Performances & scalability

Disclaimer

Robinhood v3.0 is still under development

Presented outputs, command line options, configuration parameters may change in the final release

And... It is still time to give your opinion!

Generic Policies: 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

Package	"migration" policy	"purge“ policy	"hsm_remove" policy	"rmdir" policy
robinhood-tmpfs	-	rm (old files)	-	rmdir, rm –rf
robinhood-backup	Copy to storage backend	-	rm in storage backend	-
robinhood-lhsm	Lustre HSM archive	Lustre HSM release	Lustre HSM remove	-

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, …)

Generic Policies: Overview

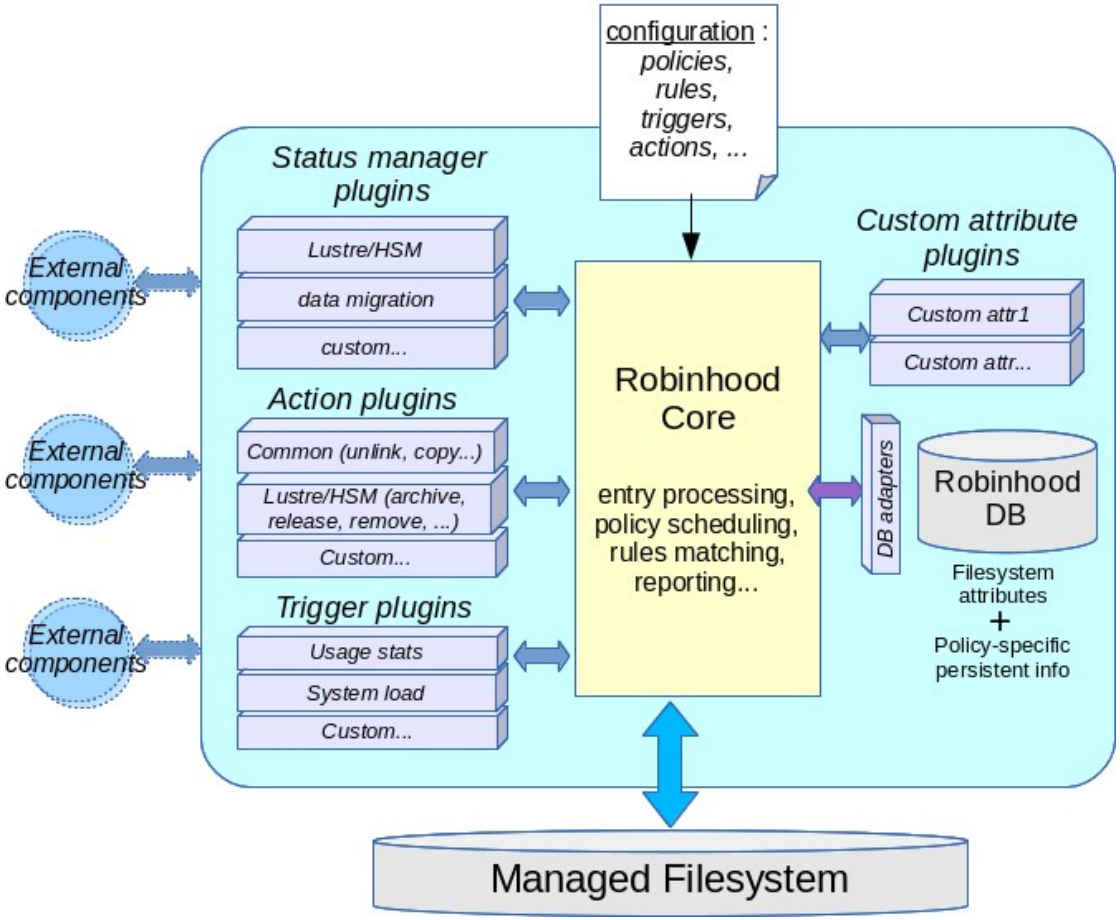
Robinhood v3 generic policies

- Robinhood core: generic policy implementation
- Specific aspects:
  - Specified by configuration (policy templates)
  - Possibly as specific plugins (dynamic libraries)
- 1 single Robinhood instance for all purposes:

Package	Generic policies
robinhood	Fully configurable

- E.g. Managing both lhsm + tmpfs modes + custom :
  - Single robinhood instance (single DB)
  - Package: `robinhood`
  - Commands: `rbh-*`
  - Config file: include `"templates/lhsm.conf"` and `"templates/tmpfs.conf"` (results in loading the related plugins)
  - Sites can define their own custom policies
  - Vendors can implement and distribute their own templates and/or plugins

Robinhood v3 Plugin-Based Architecture



Robinhood Core made generic

- 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

Plugin Types: Actions

Action plugins (v3.0)

- Actions are executed during policy runs
- Avoid forking external commands (even if it is still possible)
- Can call external APIs
- Easy specification in config file

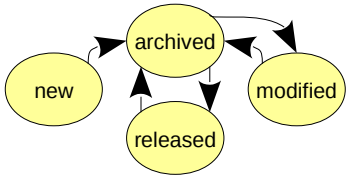
  e.g. `action = common.unlink ;`

- Shipped with robinhood v3.0:
  - `common`: common filesystem actions (copy, unlink...)
  - `lhsm`: Lustre/HSM specific actions
- Vendors/users can implement they own actions in additional plugins

Plugin Types: Status Managers

Status managers (v3.0)

- Manage specific state machines with transitions between states
- They provide:
  - Changelog records callback
  - Action callbacks
  - Specific attributes (e.g. *archive_id*, *last_archive*, ...)
  - Specific management of deleted entries
- They can interact with external components (e.g. HSM backend)
- Vendors/users can provide their own implementations as separate plugins
- Previously (v2.x): 1 single entry status for the current robinhood flavor
  - e.g. Lustre/HSM: “released”
- Now in V3: possibly one status per status manager (depends on the defined policies)
  - e.g. Lustre/HSM: “released”, Alert: “clear”, IntegrityCheck:”ok”, ...



Policy Status Reporting

Status reports

- Status reporting for each configured status manager

```
> rbh-report --status-info lhsm
lhsm.status,     type,      count,    spc_used,     volume,    avg_size
          ,      dir,          2,     8.00 KB,     8.00 KB,     4.00 KB
        new,     file,         3,           0,           0,           0
   modified,     file,         1,     4.00 KB,         15,         15
    synchro,     file,         8,     4.02 MB,     4.00 MB,   512.00 KB
   released,     file,         1,         512,     1.00 MB,     1.00 MB

> rbh-report --status-info alert
alert.status,     type,     count,    spc_used,     volume,    avg_size
       clear,      dir,         2,     8.00 KB,     8.00 KB,     4.00 KB
       clear,     file,        11,     3.02 MB,     4.00 MB,   372.37 KB
       alert,     file,         2,     1.00 MB,     1.00 MB,   512.00 KB
```

- Status support in rbh-find

```
> rbh-find -status lhsm:released
/mnt/lustre/file.1
/mnt/lustre/file.2
...
```

Custom Policy: Example

Custom policy declaration

```
declare_policy data_check {
    # manage a simple set of status: "", "ok", "failed"
    status_manager = basic;
    scope { type == file && status == "" }
    default_action = cmd("/usr/bin/mycheck.sh {path}") ;
}
```

- Then specify policy rules as usual*:

```
data_check_rules {
    ignore_fileclass = donotcheck;

    rule check_after1h {
        target_fileclass = myclass1;
        condition { creation > 1h }
    }
    ...
}
```

** "policies" renamed to "rules" in v3 for clarification*

“Legacy” Policies

Using “legacy” policies

- Modules and templates for “legacy” policies are shipped with robinhood
- You just need to “include” the right template:

```
%include “templates/lhsm.conf”
```

- Then specify policy rules as usual*:

```
lhsm_archive_rules {
    ignore_fileclass = noarchive;

    rule archive_daily {
        target_fileclass = myclass1;
        condition { last_archive> 1d or last_mod > 1d}
    }
    ...
}
```

** “migration” renamed to “lhsm_archive” in v3 templates for clarification*

Configurable Actions

- Policy actions can be specified as:
  - A **function** defined in a plugin: `<module>.<action>`

    e.g. `action = lhsm.archive ;`

    `action = common.copy ;`
  - An **external command**

    e.g. `action = cmd("/usr/bin/my_copy_wrapper.sh {path} /backup/{path}");`
- Actions can be specified at several levels:
  - In policy declaration

    e.g. in specification of "lhsm_archive" policy:

    `default_action = lhsm.archive;`
  - In policy parameters: user can override the action specified in the policy template:

    `action = cmd("/usr/bin/archive_wrapper.sh {fid}");`
  - In policy rules: allow different actions depending on targeted fileclass

```
copy_rules {
   rule copy_local {
      target_fileclass = small_files ;
      action = common.copy;
   ...
   rule copy_remote {
      target_fileclass = big_files ;
      action = cmd("dcp ...");
   ...
```

Configurable Action Parameters

- Arbitrary action parameters can be specified at multiple levels:
  - 1) As default parameters for a policy
  - 2) In policy triggers
  - 3) In policy rules
  - 4) In fileclass definitions
- If not specified, the value of a parameter is inherited from higher levels
- If specified at a lower level, the value overrides higher levels

```
migrate_parameters {
        # default parameters for the policy
        action_params {
                stripe_size = 1MB;
                stripe_count = 2;
        }
        action = cmd("lfs migrate -c {stripe_count} -S {stripe_size}");
}

migrate_rules {
   rule migrate_small {
      target_fileclass = small_files;
      action_params {
         # override default stripe_count
         stripe_count = 1;
      }
...
```

Running Policies

- 2 ways to trigger policy runs:
  - by specifying triggers in the configuration
  - by running specific command lines
- Trigger examples:

```
cleanup_trigger {
    trigger_on = ost_usage;
    check_interval = 5min;
    high_watermark_pct = 90%;
    low_watermark_pct = 80%;
}
```

```
alert_trigger {
    trigger_on = scheduled;
    check_interval = 24h;
}
```

- Trigger types:
  - **global_usage** (overall filesystem usage)
  - **ost_usage**
  - **user_usage** (volume or #entries)
  - **group_usage** (volume or #entries)
  - **scheduled/periodic** (always run at scheduled interval, with no condition on usage)
  - More to come...

Manual Policy Runs

- Running a given policy on a specific target:

  `robinhood --run=<policy> --target=<target> [limit options]`

- Implemented targets:
  - **all**: run the policy on all entries
  - **user:**<*username*>: run on entries of a given user
  - **group:**<*groupname*>: run on entries of a given group
  - **file:**<*path*>: run on a single entry
  - **class:**<fileclass>: run on entries of a given fileclass
  - **ost:**<ostidx>: run on entries striped on the given OST
  - **pool:**<ostidx>: run on entries tagged with the given OST pool

- Examples:

```
# apply the "lhsm_release" policy to entries on OST #32
robinhood --run=lhsm_release --target=ost:32

# apply the "cleanup" policy to entries of user "foo"
robinhood --run=cleanup --target=user:foo

# (re)match alerts for entries of fileclass "small"
robinhood --run=alert --target=class:small
```

Fileclass feature

Recall about fileclasses

- Robinhood allows categorizing filesystems contents according to arbitrary-defined sets called “fileclass”
- Fileclass definitions based on entry attributes:

```
Fileclass big_log {
    definition { name == “*.log”
                 and size > 100MB }
}

Fileclass admin_data {
    definition { owner == root
                 or tree == /fs/home/root }
}
```

Fileclasses in v2.x

Fileclasses in v2.x were unhandy

- Fileclasses were matched using policy rules:
  - 1 fileclass per entry per policy
  - Only matching fileclasses referenced in policies
  - Using fileclasses for monitoring only:
    needed to define a “dummy” policy

- Example of V2.x fileclass report:

```
purge_policies {
   policy purge1 {
      target_fileclass = small_files;
      target_fileclass = std_files;
      ...
   }
   policy purge2 {
      target_fileclass = big_files;
   ...
   policy default {
        ...
   }
}
```

V2.x: fileclasses in policy rules

```
> rbh-report --class-info

migr. class   ,     status,      count,   spc_used,     ...
small_files   ,        n/a,       9288,   26.18 MB,
big_files     ,        new,          7,  244.00 GB,
small_files   ,   modified,          1,    4.00 KB,
std_files     ,        new,         16,   34.78 GB,

purge class   ,     status,      count,   spc_used,    ...
small_files   ,    synchro,          7,  244.00 KB,
[ignored]     ,    synchro,   16730117,    4.12 TB,
std_files     ,   released,     349384,  534.00 GB,
[default]     ,   released,          3,    1.50 KB,
```

New fileclass implementation

In v3:

- Fileclass matching is independent from policies
  - Just need to define fileclasses to match them (unless you explicitly specify “report = no”)
- An entry can match several fileclasses
  - Allow mapping filesystem contents using various criteria
    e.g. *small vs. big*
    log vs. *data* vs. ...
- Fileclass reports indicate fileclass combinaisons:

```
> rbh-report --class-info

fileclass            ,     count,   spc_used,     ...
small+log+admin      ,      9288,   26.18 MB,
big+log+user         ,         7,  244.00 GB,
small+user           ,  16730117,    4.12 TB,
small+user+important,    349384,  534.00 GB,
std+user+tmp         ,    263273,  344.65 GB,
```

- Fileclasses can still be used in policy rules

```
target_fileclass = …
```

Robustness & code quality

- SQL requests in *ListMgr* now built using glib2 strings
  - Eliminate risks of overflows for large requests (e.g. wide stripes)
  - Clean many Coverity warnings
  - Lighten the code
- Automated Valgrind validation
  - The whole test suite runs under valgrind and checks valgrind errors
- Factorized duplicate code patterns
  - V2.5: 55k lines of code
  - V3.0: 52k lines of code
- Modernized code

  - stdbool, callback-based iterators, lower case source files and functions, ...

```
[b_3.0]> git diff -r 2.5.5 --stat

417 files changed, 44852 insertions(+), 46653 deletions(-)
```

What About Performance?

- Most performance improvements developped in branch v3 have been backported to release v2.5.5
- Other major improvements scheduled for next releases 3.1, 3.x...

When Can I Play?

Current status

- Most of the development is done
- Current status is almost “alpha” (for *lhsm* and *tmpfs* features)
- You can start playing with it by getting branch “b_3.0” from git

Still to be done

- Legacy “backup” mode not yet functional
- Undelete and disaster recovery for Lustre/HSM
- A few minor bugs to be fixed
- User documentation
- V2.5 to v3 upgrade helpers

Beta is targeted in Q4 2015
Final version Q1 2016

Release Cycle

V3 release cycle

- V3.0 is a first step, but we plan other major changes in the short-term (see Henri's talk)
- It is expected that versions 3.1, 3.2... will follow quickly

Conclusion

Robinhood v3 in a nutshell

- Plugin-based architecture
  - More generic and powerful robinhood core
  - Allows integration of vendor-specific or site-specific modules
- Easily implement new policies just by writing a few lines of configuration:
  - OST rebalancing
  - Pool-to-pool data migration
  - Data integrity checks
  - Trash can mechanism
  - Massive data conversion
  - ...
- Improved robustness and code quality
- V3.0 is not the last stop!
  - Other major improvements to come in the very next releases!

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

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