FROM RESEARCH TO INDUSTRY

www.cea.fr

RobinHood v3

Integrity check &

thoughts on local VFS changelogs

Robinhood User
Group 2016

Dominique Martinet <dominique.martinet@cea.fr>

19 SEPTEMBER 2016

Robinhood v3 as announced last year

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
  - …

Where are we now?

robinhood@home

- Been using robinhood v3 to checksum files at home since Feb
  - Hasn't eaten my data yet
    - sadly (fortunately?) no corruption found yet
  - More usable than my old manual yearly-ish checksum runs
    - What's wrong with `find /mnt/data -exec kludgy_checksum_script.sh {} +` ?
  - Might as well give examples on tools available

- Thoughts on what could be improved next: “VFS changelogs”
  - working with VFS handles where we can
  - VFS notification mechanisms

DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

Playing at home with checksums

The checker

New status manager: checker

- Three status: '', 'ok' and 'failed'
- Manages three attributes:
  - output – stdout of checker on success (255 first chars)
  - last_check – unix time of last check
  - last_success – unix time of... last success
- Attributes are kept in DB and available for use in policy commands, report, find...
- Named checker, but very versatile: output can be anything
  - Run “file” as check command, get statistics of what kind of files we manage
  - Use checksum as a hash and look for duplicate files
  - Statistics on multimedia files (size, duration, bitrate...)
- Just 150 lines of code!

DE LA RECHERCHE À L'INDUSTRIE

The checker

Provided checksum script: rbh_cksum.sh

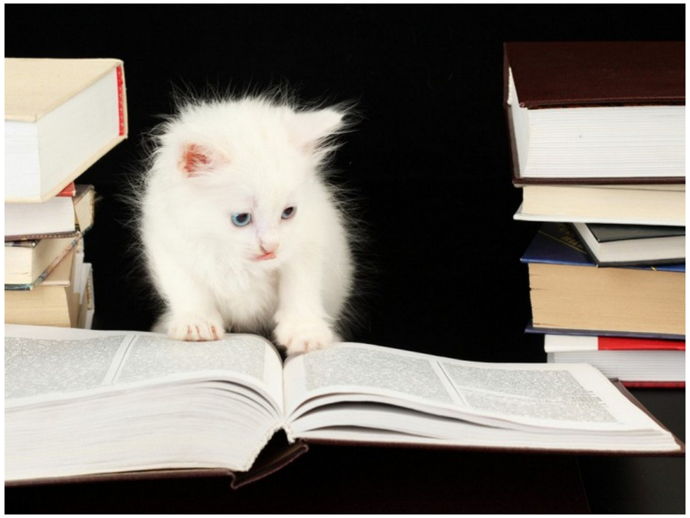
- Compute checksum with your preferred tool (keeps first word from stdout)
- Has file changed? Use Lustre's *data_version* if able, or “*mtime-size*”
- Wails if checksum changed with no apparent modification,
- Or purrs and possibly stores output in xattr as well as robinhood's DB

```
> less /etc/sysconfig/rbh_cksum

RBH_CKSUM_CMD=sha1sum

RBH_CKSUM_DV_CMD='lfs data_version'
RBH_CKSUM_DV_CMD='stat -c "%Y-%s"'

RBH_CKSUM_XATTR=user.sha1sum
```



Configuration

Generic policy

```
define_policy checksum {
  status_manager = checker;
  scope { type == file }
  default_lru_sort_attr = last_check;
  default_action =
    cmd("/usr/sbin/rbh_cksum.sh '{output}' '{path}'");
}
```

packaged /etc/robinhood.d/includes/check.inc

```
checksum_trigger {
  trigger_on = periodic;
  check_interval = 12h;
}

checksum_parameters {
  nb_threads = 2;
  max_action_volume = 250GB;
  max_action_count  = 350000;
  suspend_error_pct = 50%;
  suspend_error_min = 100;
}
```

triggers/params close to v2 syntax, per policy

```
%include “includes/check.inc”

fileclass never_checked {
  definition { checksum.last_success == 0 }
  report = no;
}


checksum_rules {
  ignore { last_mod < 6h }
  ignore { last_check < 45d }

  rule never_checked {
    target_fileclass = never_checked;
    condition = true;
  }

  rule default {
    condition = true;
  }
}
```

example rules (/etc/robinhood.d/templates/example.conf)

Running checker

Service

- Usually something like:
  - *robinhood --scan --run=all*
  - *robinhood --readlog --run=all*
- Reads /etc/sysconfig/robinhood for options
- Can have multiple instances of robinhood running
  - e.g. one with *--scan* and one per policy
  - lets you fiddle with policies without triggering a new scan every restart
- Despite all that's said, systemd unit files are nice compared to old init script
  - 332 lines vs. 9 lines

Running checker

One-shot

- Same commands given earlier can be run manually
- Can add targets & more options
  - user, file, class, ost... see --help!

```
> robinhood --run='checksum(target=file:/mnt/data/a/build/robinhood/src/robinhood/robinhood)' -I

2016/09/01 09:13:17 [25512/2] checksum | Checking policy rules for entry
'/mnt/data/a/build/robinhood/src/robinhood/robinhood'
2016/09/01 09:13:17 [25512/2] checksum | Executing policy action on: 3D072B/6906378 (
/mnt/data/a/build/robinhood/src/robinhood/robinhood)
2016/09/01 09:13:17 [25512/2] cmd_stderr | /mnt/data/a/build/robinhood/src/robinhood/robinhood: new
cksum: 1472577979-2516736:64b54f144e9a1802829dc7b28090e27af9759b05
2016/09/01 09:13:17 [25512/2] checksum | Policy run summary: time=01s; target=entry
'/mnt/data/a/build/robinhood/src/robinhood/robinhood'; 1 successful actions (1.00/sec); volume: 2.40 MB
(2.40 MB/sec); 0 entries skipped; 0 errors.


> robinhood --run='checksum(target=file:/mnt/data/a/build/robinhood/src/robinhood/robinhood)' -I

2016/09/01 09:14:05 [25561/2] cmd_stderr | /mnt/data/a/build/robinhood/src/robinhood/robinhood: cksum
OK: 1472577979-2516736:64b54f144e9a1802829dc7b28090e27af9759b05
```

Summary of checker

- rbh-report can give a summary of the checker's activity
  - split by status
  - can filter on a given class with *-C <class>* or other usual options
- Example summary report

```
> rbh-report --status-info checksum
Using config file '/etc/robinhood.d/data.conf'.
checksum.status,     type,      count,      volume,    spc_used,    avg_size
               ,  symlink,      24465,     1.36 MB,    21.13 MB,          58
               ,      dir,     254934,     3.09 MB,     1.02 GB,          13
               ,     file,        109,     8.91 GB,     8.92 GB,    83.71 MB
               ,     fifo,         10,           0,     5.00 KB,           0
               ,     sock,          5,           0,     2.50 KB,           0
            ok,     file,    2447721,     4.98 TB,     4.99 TB,     2.13 MB
        failed,     file,          0,           0,           0

Total: 2727244 entries, volume: 5483542728084 bytes (4.99 TB), space used:
5502279452672 bytes (5.00 TB)
```

rbh-find

Find files with a given state

- Can dump all files with *checker:status* syntax

- Obviously can combine with all the usual rbh-find options:
  - path to search
  - -mtime [-|+]<val>[s|m|h|d|y]
  - -size [-|+]<val>[K|M|G|T]

- Example output:

```
> rbh-find -status checksum:failed -type f -lsstatus

3D072B/65537 file 1048576 checksum:failed /mnt/data/tests/checkme
```

Have fun with printf

- Can output **anything**

```
> rbh-find -status checksum:ok -printf "%Rm{checksum.output} %p\n" -type f
-name policy_run.c

1467457422-93459:3f22725a868a702e1b513b40dc612e3904277590
/mnt/data/a/build/robinhood/src/policies/policy_run.c
```

```
 > rbh-find --help
[...]
            %p   Full file name
            %Rc  File class
            %Rf  Lustre FID
            %Rm  Status manager module attribute, with the name specified
between curly bracket. The name is the status manager module name, followed by
a dot, followed by the attribute name. For example: %Rm{lhsm.archive_id}.
            %Ro  Lustre OSTS
            %Rp  Lustre parent FID
[...]
```

Wishlist

Possible improvements after running a few months

- Externally trigger a run with specific non-default conditions
  - Currently need to define a new rule or fileclass for a one-shot run
    - for example, run once on all the failed entries after fixing checker

```
robinhood --run='checksum(condition={ status == failed })'

robinhood --run='checksum(target=more_checks,condition={ last_check < 15d })'
```

- More complex rules
  - Can't compare two attributes
    - re-run when file is modified (simulate “dirty” state)

```
rule recheck {
  condition { last_mod < last_check } # caution here last_x is time since last x, not timestamp
}
```

Wishlist

Possible improvements after running a few months

- Multiple rule-targeted triggers
  - Build multiple policy_run schedules for different set of rules
  - Allows better optimisation (building specialized DB queries)

```
checksum_rules {
  ignore { last_mod < 6h }
  ignore { last_check < 45d }

  rule more_checks {
    target_fileclass = more_checks;
    condition = true;
  }

  rule never_checked {
    target_fileclass = never_checked;
    condition = true;
  }

  rule default {
    condition { last_check < 180d }
  }
}
```

```
checksum_trigger {
  trigger_on = periodic;
  check_interval = 12h;
  trigger_rule = more_checks, never_checked;
}

checksum_trigger {
  trigger_on = periodic;
  check_interval = 15d;
  trigger_rule = default;
}
```

Wishlist

Tools improvements

- checker script could probably be improved
  - vmtouch: evict from cache if file wasn't already cached
  - handle partial lustre paths e.g. <dirfid>/foo/bar
- Contributions welcome ! (probably)

DE LA RECHERCHE À L'INDUSTRIE

Alternatives to changelogs for local filesystems

www.cea.fr

Motivation

Changelogs are awesome!

- Full rescans are slow
- Partial rescans are not enough
  - cannot tell if missing files were moved or deleted
  - slow anyway and/or doesn't fit all usages (*--no-gc*)
- Checker can read file from cache if triggered shortly after file creation
  - Don't trust the first disk write

DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

VFS handles

VFS handles: tracking file movements

name_to_handle_at, open_by_handle_at

- Persistent handles
  - Can store them in the database
- Similar to *.lustre/fid/<fid>*
  - Actually works with lustre too!
    - Lustre handle is binary fid + handle type/size (constant on lustre)
    - Other filesystems usually have inode number + generation id as “fid”
- Poor man's fid2path: open and check path in */proc/self/fd/*
  - No hard link list
- Easy to check if files moved or deleted if we get ENOENT
- ⚠ Does not work with all FS (e.g. NFS)

DE LA RECHERCHE À L'INDUSTRIE

VFS events

www.cea.fr

Monitoring file system events

inotify

- Set up “watch directories” dir one at a time
  - New directories need to manually be added to the watch
    - Race conditions
    - Scalability issues

- Complete set of events
  - data access and modifications (`close_nowrite, close_write`)
  - `create, delete, move_self, moved_from, moved_to`
  - “attrib” (owner, mode, timestamp, xattr and link count)

- ⚠ move_from does not give fd nor new filename (gives old name)
  - We can work around that with vfs handles

DE LA RECHERCHE À L'INDUSTRIE

Monitoring file system events

fanotify

- Whole filesystem level
  - set up once for the mount point
- Enumerates data-related events
  - accesses (`open/read`)
  - modify (`write/close, close_write or close_nowrite`)
  - only gives an open fd to the files (path through /proc/self/fd)
- But. . . Does not catch metadata events
  - No rename/unlink
- (fun fact: can have the kernel ask userland for permission for other processes to open files)

Hybrid solution

The best of both worlds

- fanotify is more suited for whole filesystem watching
- which does not mean we can't **also** use inotify on a list of configurable directories to catch moves/unlinks
- Only works for simple usage patterns, but good enough if policy commands validate path
  - already do *lstat()* before run
  - easy enough to try *open_by_handle_at()* and get new path on failures
  - (or could pass an already open fd to said commands like generic copytool!)

Thank you for your attention !

Questions ?