RobinHood Policy Engine

http://robinhood.sf.net

Quick Tour

2011/05/23

v2.3.0
The issue with large filesystems

**Common needs / usual solutions:**

- Space usage accounting
  - Per user, per group → quotas
  - Per project, per directory, ...
    → du
    → `find /fs -ls | acct.sh`

- FS content profiling
  → `find /fs -ls | profile.sh`

- Purge old unused files
  → `find /fs -atime +30 -delete`

- Detect « bad » FS usage, find specific entries
  → `find /fs <criteria>`

- Lustre: balance OST usage
  → `lfs df`
  → `lfs find -ost ...`

---

http://robinhood.sf.net

Robinhood PolicyEngine: Quick Tour
The issue with large filesystems

- **Scanning (find, du, ...) becomes endless as filesystems grow**
  - FS size grows quicker than avg file size
    - more metadata, larger namespaces, longer scans…
  - Real life cases:
    - 1 million => 45 min
    - 20 millions => 15 hours
    - 200 millions => 6 days

- **In most cases, it needs a new endless scan for each specific action**

- **Sometimes, it’s too late:**
  - when the FS is full, you can’t wait hours for `find` to end

- **Also, needs to write a specific script for each specific usage**
**Principle: scan sometimes, query often**

- Robinhood querying tool
- SQL

**Build-in features / policies:**
- purge files by LRU
- data archiving
- soft rm
- customizable alerts

- 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**
(no more scan is needed)
RobinHood performance

- **All actions are performed in parallel**
  - Multi-threaded scan
  - Multi-threaded purge, data archiving, …

- **Scan performance** (Lustre 2.0):

  ![Scan performance diagram]

  - Scanning a FileSystem with 1.5M files

  ![Diagram showing scan duration in minutes]

<table>
<thead>
<tr>
<th>Method</th>
<th>Scan duration (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>find -ls</td>
<td>60</td>
</tr>
<tr>
<td>robinhood 2 threads</td>
<td>42</td>
</tr>
<tr>
<td>robinhood 4 threads</td>
<td>24</td>
</tr>
</tbody>
</table>

http://robinhood.sf.net  Robinhood PolicyEngine: Quick Tour
Stats & Accounting

- **Per user, per group:**
  - `rbh-report -u foo* --csv`

<table>
<thead>
<tr>
<th>user</th>
<th>count</th>
<th>spc_used</th>
<th>avg_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo1</td>
<td>11000</td>
<td>11534336000</td>
<td>10485760</td>
</tr>
<tr>
<td>foo2</td>
<td>101398</td>
<td>3780071239680</td>
<td>37286674</td>
</tr>
</tbody>
</table>

- **Per user AND per group:**
  - `rbh-report -u foo --csv --split-user-groups`

<table>
<thead>
<tr>
<th>user</th>
<th>group</th>
<th>count</th>
<th>spc_used</th>
<th>avg_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>proj1</td>
<td>1542</td>
<td>114336000</td>
<td>74147</td>
</tr>
<tr>
<td>foo</td>
<td>proj2</td>
<td>1013</td>
<td>3780071239</td>
<td>3731560</td>
</tr>
</tbody>
</table>

- **FS content summary**
  - `rbh-report -i --csv`

<table>
<thead>
<tr>
<th>type</th>
<th>count</th>
<th>spc_used</th>
<th>avg_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>130542</td>
<td>534700032</td>
<td>4096</td>
</tr>
<tr>
<td>file</td>
<td>1256830</td>
<td>378007123900</td>
<td>300700</td>
</tr>
<tr>
<td>symlink</td>
<td>1013</td>
<td>30717</td>
<td>30</td>
</tr>
</tbody>
</table>

- **Possibly filter by directory:**
  - `rbh-report -u foo -P '/fs/one_dir/dir*'`
Other built-in reports:
- `rbh-report --top-users | --top-size`
- `rbh-report --dump-user='foo24'`
- `rbh-report --dump-ost=22`
- See « `rbh-report --help` » ...

Incoming features for accounting:
- Accounting per project (with arbitrary definition)
  - Ex.
    ```
    project1 { tree == '/fs/dir.1' and group == g1 }
    project2 { tree == '/fs/dir.1' and group == g2 }
    ```
  - Reports would split usage in projects:
    ```
    user1/project1, user2/project1, user1/project2, user2/project2...
    ```
- `find` and `du` clones querying robinhood DB
- Web gui (charts, browsable reports)
Customizable Alerts

- **Alert on anormal filesystem entries:**
  - Flexible, attribute-based alert definitions:
    ```
    Alert large_file_in_bad_place {
      type == file
      and size > 1TB
      and tree != "/fs/big_files"
    }
    ```

- **Quota-based alerts**
  - Send mail if a user/group exceeds a given threshold:
    ```
    trigger_on = user_usage(foo*,bar*);
    high_threshold_vol = 20TB;
    notify = TRUE;
    ```

- **Sends mail to admin / write alert to a specific log file**

- **Incoming feature for alerts:**
  - Send alert to file owner (with customized message)
Purge policies: overview

- **Purge files from the least recently used**
  - different from: `find -atime +30 -delete`
  - no need to purge all files > x days, if not necessary

- **Purge can be triggered in different manners:**
  - By the admin (command line)
    - Purge filesystem until disk usage is back to 80%, purge OST#3 until its usage is back to 85%, …
  - Periodically
    - apply purge policies every hour, …
  - On usage threshold
    - if FS usage > 90%, purge data until usage = 89%
    - If OST usage > 85%, purge data in this OST until its usage = 80%

- **Purge policies:**
  - Can purge some files earlier than others
  - Rules to determine if a file can be purged:
    - based on file properties (size, last access, xattrs…)
  - Files can be whitelisted

http://robinhood.sf.net

Robinhood PolicyEngine: Quick Tour
Purge triggers

Purge trigger examples:

- **FS usage:**
  ```
  purge_trigger {
    trigger_on = global_usage;
    high_threshold_pct = 90%;
    low_threshold_pct = 85%;
    check_interval = 10min;
  }
  ```

- **OST usage:**
  ```
  purge_trigger {
    trigger_on = ost_usage;
    high_threshold_pct = 90%;
    low_threshold_pct = 85%;
    check_interval = 10min;
  }
  ```

- **User usage:**
  ```
  purge_trigger {
    trigger_on = user_usage(foo*, bar*);
    high_threshold_vol = 10TB;
    low_threshold_vol = 9TB;
    check_interval = 1d;
    notify = TRUE;
  }
  ```
Basic purge policy:

- **Ignore** statement for whitelisting entries
- **Default** policy case
- **Required** condition for purging entries
  (doesn’t mean all matching entries will be removed)

```plaintext
purge_policies {
  ignore { tree == /fs/save_me
    or size == 0
    or xattr.user.pin == 1 }

  policy default {
    condition { last_access > 1h }
  }
}
```
FileClasses

● Fileclass definitions:

FileSets
{
  FileClass Small_files { definition { size < 10MB } } 
  FileClass Medium_files { definition { size >= 10MB and size < 1GB } } 
  FileClass System_Logs {
    definition {
      name == '*.log'
      and owner == 'root'
    }
  }
}

FileClass SmallLog { definition { System_Logs inter Small_files } } 
FileClass BigLog { 
  definition { 
    System_Logs inter 
    not ( Small_files union Medium_files )
  }
}
}
Purge Policy using Fileclasses

- **Purge policy using fileclasses**
  - Can apply different policies for each fileclass

```cpp
purge_policies {
  ignore_fileclass = Small_files;
  ignore_fileclass = class2;

  policy purge_7d {
    target_fileclass = BigLog;
    target_fileclass = UserFiles1;

    condition { last_access > 7d }
  }
  policy purge_1d {
    target_fileclass = UserFiles2;

    condition { last_access > 1d }
  }
  policy default {
    condition { last_access > 12h }
  }
}
```
Bonus: Profiling FS content

- FileClasses can be used for profiling FS content
  
  **rbh-report --class-info**

<table>
<thead>
<tr>
<th>FileClass</th>
<th>count</th>
<th>spc_used</th>
<th>avg_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class_X</td>
<td>1100</td>
<td>1153433600</td>
<td>10485760</td>
</tr>
<tr>
<td>Class_Y</td>
<td>10139</td>
<td>378007123968</td>
<td>37286674</td>
</tr>
</tbody>
</table>

- List all entries in a FileClass
  
  **rbh-report --dump --filter-class='class_*'**

  /fs/dir_1/.../file.1
  /fs/dir_1/.../file.2
  /fs/dir_2/.../file.x
  /fs/dir_n/.../file.y
Robinhood Flavors

- **robinhood-tmp_fs_mgr (GA, stable)**
  - Management of scratch filesystem
  - Policies: purge (rm), rmdir
  - Commonly used mode
  - Support: all Posix FS, Lustre 1.x, Lustre 2.x

- **robinhood-lhsm (coming soon)**
  - PolicyEngine for Lustre-HSM binding (Lustre v2.x?)
  - Lustre manages: file state (dirty bit…), file recall, release, …
  - Robinhood policies: schedule archiving, releasing files, delayed removal in backend
Data archiving (Lustre-HSM)

- **Migration (archiving) policy:**
  - Similar to purge policies (using fileclasses)
  - Can specify hints for copy command
    - E.g. target class of service in storage system, ...

- **Example:**

  Migration_policies {
    ignore { size < 1KB or tree == "/fs/tmp/logs" }

    policy archive_user_files_A
    {
      target_fileclass = user_files_A ;
      condition {
        last_mod > 3h
      }
      archive_num = 2; # target storage system
      migration_hints = "file_family_width=3,priority=3";
    }
  }

  policy default {
    ...
  }

http://robinhood.sf.net
Limitations & future improvements

● Before Lustre 2.1, scan is needed
  ■ Multi-threading increases scan speed, but it is still too long with millions/billions of files (can take days)
  ■ Makes it hard to have up-to-date info in DB
  ■ Possible improvements: low-level scan (e.g. read MDT device), FS bulk scan feature, …

● robinhood v2.3 (just released)
  ■ Optimizations for common accounting needs (user, group, …)
    => make user/group usage reporting instantaneous \(O(N) \rightarrow O(1)\)
  ■ But some reports, with customized filters, are difficult to optimize
    ➢ E.g. need to SELECT SUM(size)… on the whole table
    ➢ => takes ~several minutes with 100 million entries

● Future optimizations for other common needs:
  ➢ compute ‘du’ efficiently on any directory of the filesystem

● Other future evolution: NoSQL database
About the Project

- **Robinhood PolicyEngine is OpenSource**

- **Website:**
  - [http://robinhood.sf.net](http://robinhood.sf.net)
  - Downloads (RPMs and tgz)
  - Ticket tracking system
  - Wiki

- **Mailing lists:**
  - [robinhood-support@lists.sourceforge.net](mailto:robinhood-support@lists.sourceforge.net)
  - [robinhood-news@lists.sourceforge.net](mailto:robinhood-news@lists.sourceforge.net)
  - [robinhood-devel@lists.sourceforge.net](mailto:robinhood-devel@lists.sourceforge.net)

- **Git repository**
  - [http://robinhood.git.sourceforge.net](http://robinhood.git.sourceforge.net)