Giving easy user access to Lustre jobstats and Robinhood informations

By Simon Guilbault
Introduction

● Improving jobstats
  ○ Information was not available to the end users, or aggregated at the user or application level

● Diskusage_explorer
  ○ Giving a easy way for the users to find “where” their quotas is taken in their project
Improving jobstats

And giving that information to the users
lustre_exporter by itself

- [https://github.com/HewlettPackard/lustre_exporter](https://github.com/HewlettPackard/lustre_exporter)
- With Prometheus
- Great to capture stats per job ID
  - IOPS and bandwidth
  - Cannot aggregate easily per user or group
    - Problem with single core jobs

```
lustre_job_write_bytes_total{
  component="ost",
  jobid="18511257",
  target="lustre03-OST0007"}
```
Intercept and improve metrics

- Lustre servers do not have detailed job information
  - Fetch missing information in jobstats from slurm/ldap
  - Python script as a proxy

https://github.com/guilbaults/lustre_exporter_slurm
Adding tags (slurm jobs)

From $SLURM_JOB_ID

lustre_job_write_bytes_total{
  component="ost",
  jobid="18511257",
  target="lustre03-OST0007",
  fs="lustre03",
  user="user1",
  account="group1"
}
Adding tags (logins, DTNs ...)

From procname_uid

```
lustre_job_write_bytes_total{
  component="ost",
  jobid="tar.1000",
  target="lustre02-OST0000",
  fs="lustre02",
  application="tar",
  user="user1"
}
```
Pattern on login nodes

\[
\text{sum by (user) (rate(lustre_job_stats_total\{application="touch"\}[5m]))}
\]
Grafana

- Detailed information is now available to sysadmins
  - Analysts and users do not have access to Grafana
  - Can’t easily restrict a user to their own stats
  - Sending snapshots in ticket as a “proof” of bad IO pattern
  - Fixed the worse top 10 users
Public dashboard

- Stop gap solution
- https://dashboard.beluga.calculquebec.ca/filesystems.html
- Static pages with Jekyll
- Grafana renderer
  - Replaced by Matplotlib
- Translation
User portal

- Recent jobs (SlurmDB)
- Stats per user, account and job (Prometheus)
  - Filesystem performance
  - CPU, GPU, Memory
    - Allocated and actual uses
- Filesystems quotas and HSM state (Robinhood)
- Built with Django (Python)
  - Using Shibboleth's authentication
  - Translation framework
  - “The web framework for perfectionists with deadlines.”
Recent jobs

- Breakdown per job
- Live job stats

### Your jobs

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Status</th>
<th>Job name</th>
<th>Submit time</th>
<th>Start time</th>
<th>End time</th>
<th>Asked time</th>
<th>Used time</th>
</tr>
</thead>
<tbody>
<tr>
<td>185156987</td>
<td>Running</td>
<td></td>
<td>49 minutes ago</td>
<td>44 minutes ago</td>
<td></td>
<td>106.0m</td>
<td></td>
</tr>
<tr>
<td>185156892</td>
<td>Running</td>
<td></td>
<td>50 minutes ago</td>
<td>49 minutes ago</td>
<td></td>
<td>106.0m</td>
<td></td>
</tr>
<tr>
<td>185156891</td>
<td>Running</td>
<td></td>
<td>55 minutes ago</td>
<td>44 minutes ago</td>
<td></td>
<td>106.0m</td>
<td></td>
</tr>
<tr>
<td>185165346</td>
<td>Running</td>
<td></td>
<td>an hour ago</td>
<td>58 minutes ago</td>
<td></td>
<td>106.0m</td>
<td></td>
</tr>
<tr>
<td>185153355</td>
<td>Cancelled</td>
<td></td>
<td>an hour ago</td>
<td>59 minutes ago</td>
<td></td>
<td>72.0h</td>
<td></td>
</tr>
<tr>
<td>185160668</td>
<td>Cancelled</td>
<td></td>
<td>an hour ago</td>
<td>55 minutes ago</td>
<td></td>
<td>72.0h</td>
<td></td>
</tr>
<tr>
<td>185091221</td>
<td>Failed</td>
<td></td>
<td>3 hours ago</td>
<td></td>
<td>48.0h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>185279774</td>
<td>Failed</td>
<td></td>
<td>3 hours ago</td>
<td></td>
<td>48.0h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>184661564</td>
<td>Running</td>
<td></td>
<td>13 hours ago</td>
<td>13 hours ago</td>
<td></td>
<td>48.0h</td>
<td></td>
</tr>
<tr>
<td>184980559</td>
<td>Running</td>
<td></td>
<td>16 hours ago</td>
<td>15 hours ago</td>
<td></td>
<td>48.0h</td>
<td></td>
</tr>
<tr>
<td>184683410</td>
<td>Failed</td>
<td></td>
<td>20 hours ago</td>
<td>17 hours ago</td>
<td></td>
<td>48.0h</td>
<td>0.1h</td>
</tr>
<tr>
<td>184581466</td>
<td>Failed</td>
<td></td>
<td>1 day, 3 hours ago</td>
<td>1 day, 3 hours ago</td>
<td>14 hours ago</td>
<td>48.0h</td>
<td>12.8h</td>
</tr>
<tr>
<td>184555021</td>
<td>Failed</td>
<td></td>
<td>1 day, 3 hours ago</td>
<td>1 day, 3 hours ago</td>
<td>1 day, 3 hours ago</td>
<td>48.0h</td>
<td>5.8h</td>
</tr>
<tr>
<td>184449166</td>
<td>Failed</td>
<td></td>
<td>1 day, 11 hours ago</td>
<td>1 day, 11 hours ago</td>
<td>1 day, 7 hours ago</td>
<td>48.0h</td>
<td>4.4h</td>
</tr>
<tr>
<td>184455883</td>
<td>Failed</td>
<td></td>
<td>1 day, 15 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>48.0h</td>
<td>38.2h</td>
</tr>
<tr>
<td>184419677</td>
<td>Failed</td>
<td></td>
<td>1 day, 18 hours ago</td>
<td>1 day, 18 hours ago</td>
<td>an hour ago</td>
<td>48.0h</td>
<td>38.0h</td>
</tr>
<tr>
<td>184415530</td>
<td>Failed</td>
<td></td>
<td>1 day, 18 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>48.0h</td>
<td>1.0h</td>
</tr>
<tr>
<td>184429687</td>
<td>Failed</td>
<td></td>
<td>1 day, 18 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>23 hours ago</td>
<td>48.0h</td>
<td>15.8h</td>
</tr>
<tr>
<td>184423035</td>
<td>Failed</td>
<td></td>
<td>1 day, 22 hours ago</td>
<td>1 day, 22 hours ago</td>
<td>1 day, 21 hours ago</td>
<td>24.0h</td>
<td>29.3h</td>
</tr>
<tr>
<td>184197720</td>
<td>Completed</td>
<td></td>
<td>1 day, 22 hours ago</td>
<td>1 day, 22 hours ago</td>
<td>1 day, 15 hours ago</td>
<td>48.0h</td>
<td>30.5h</td>
</tr>
</tbody>
</table>
Filesystem performance

A user can see their own use
CPU, memory and GPU
Quotas and HSM states

● Breakdown per user in a group (uid, gid)
  ○ Not per directory

● HSM (tape) status
diskusage_explorer

Robinhood and `duc`
Typical tickets

● Who filled my group quota?
  ○ lfs quota -u does not handle a user in multiple groups
  ○ Robinhood can provides this breakdown per uid, gid

● Where?
  ○ Asking the user with millions of files in the group
    ■ Good luck with lfs find or du
Current tools

446M INODES

du
lfs
find
Need a UI for the users

- `ncdu` got a UI, but need to scan the FS or load a gigantic pre-scanned JSON
- Found `duc` as an alternative
  - "Duc stores the disk usage in an optimized database, resulting in a fast user interface. No wait times once the index is complete."
User point of view with duc

$ diskusage_report

[... lfs quota output are here ...]

Disk usage can be explored using the following commands:

diskusage_explorer /home/sigui4           (Last update: 2021-04-21 13:17:48)
diskusage_explorer /scratch/sigui4       (Last update: 2021-04-21 13:20:06)
diskusage_explorer /project/def-sigui4   (Last update: 2021-04-21 09:28:54)
diskusage_explorer /nearline/def-sigui4          (Last update: 2021-04-21 14:01:50)

- diskusage_explorer will launch `duc` with the correct database (`duc ui --database={}`)
- The user does need to search where the database is stored on the FS.
CLI UI example

- Can display by actual size, apparent size and inodes
  - compression, HSM and sparse file -> actual != apparent
  - Switching directory take a few ms
duc

- [http://duc.zevv.nl/](http://duc.zevv.nl/)
- Intended to scan a local FS
- Some previous attempts to scan Lustre and GPFS
  - [https://github.com/zevv/duc/issues/259](https://github.com/zevv/duc/issues/259)
  - [https://github.com/zevv/duc/issues/180](https://github.com/zevv/duc/issues/180)
Robinhood database to the rescue

- Robinhood already have all the required information
  - Not directly accessible by users
  - Does not provides a aggregation per directory
- Changelogs keep the MySQL DB up to date with the changes on the FS
- (lazy size on MDT should also work)
DUC database

- Support multiple key-value DB
  - tokyocabinet, kyotocabinet, leveldb, lmdb?
- Also support sqlite
  - Fast enough for us, easier to install and debug
- Store information of each directory in a binary format
  - Variable length integer to save space...
robinhood2duc

- Depth-first search using the tree in MySQL
  - Only using the metadata stored by Robinhood
- Produce a sqlite file for each directory (including every subdirectories)
  - Every user in project have access, stored on Lustre
  - 1h for a project with 18M inodes
  - Updated multiple times per day with a crontab
    - Run with gnu-parallel
DB server metrics
Robinhood DB server

- Overkill, modified login node
  - 40 cores, use less than 20%
  - 196 GB of ram, use all of it
  - 2 NVMe of 1.6TB (PM1725b)
    - Use 547GB for 450M inodes
      - ZFS compress it down to 225GB
    - Average 1000 IOPS, 100MB/s
- (Lustre FS does on average 20k IOPS, can peak at a few 100k IOPS depending on the jobs)
Git repo

https://github.com/guilbaults/robinhood2duc
Conclusion

- Users can now "see" what resources they are using
  - Job level stats, and various aggregations
- "where" their files are in the filesystem