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

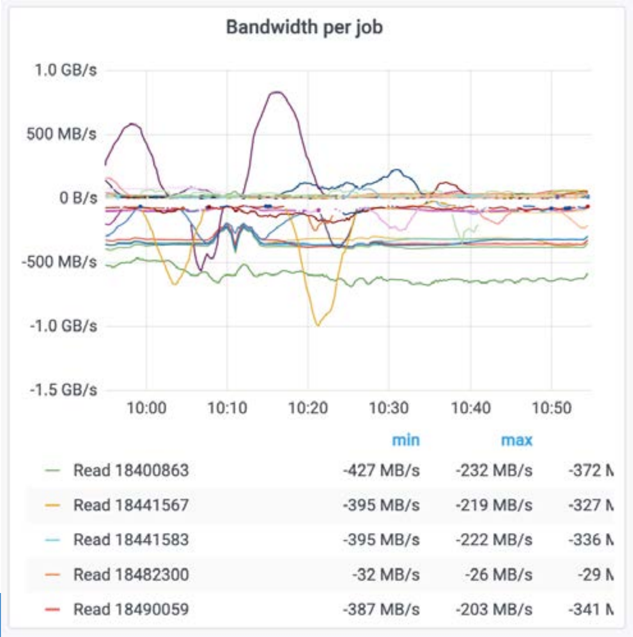
Improving jobstats

And giving that information to the users

lustre_exporter by itself

- https://github.com/HewlettPackard/lustre_exporter
- 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"}
```

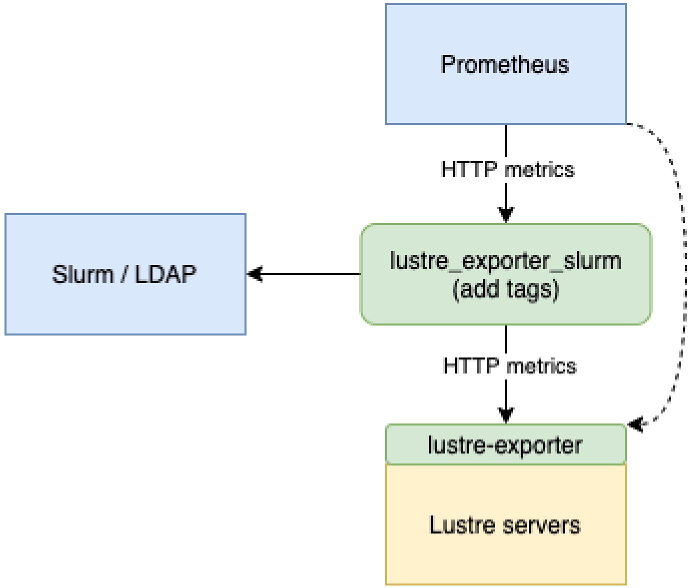


	min	max	
Read 18400863	-427 MB/s	-232 MB/s	-372
Read 18441567	-395 MB/s	-219 MB/s	-327
Read 18441583	-395 MB/s	-222 MB/s	-336
Read 18482300	-32 MB/s	-26 MB/s	-29
Read 18490059	-387 MB/s	-203 MB/s	-341

Intercept and improve metrics

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

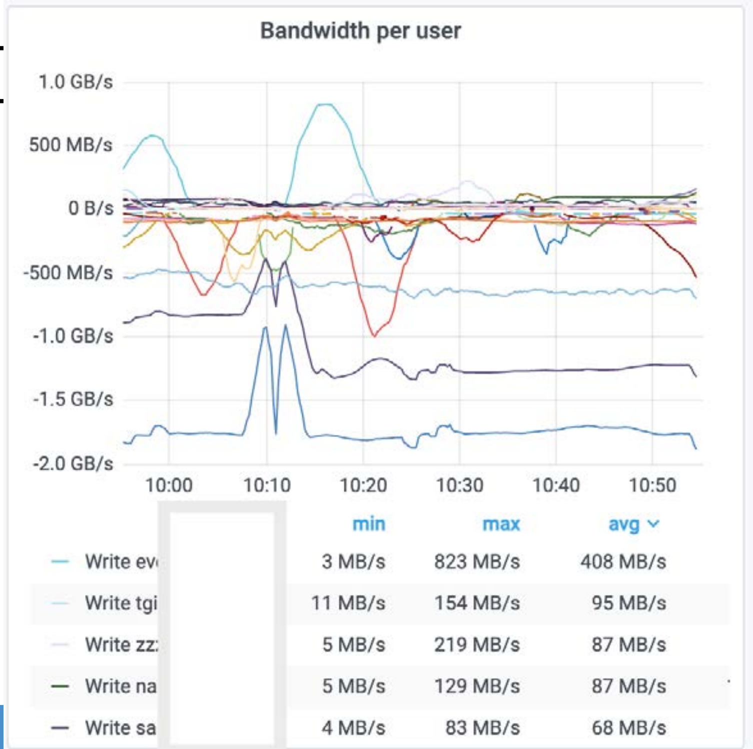
https://github.com/guilbaults/lustre_exporter_slurm



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"}
```

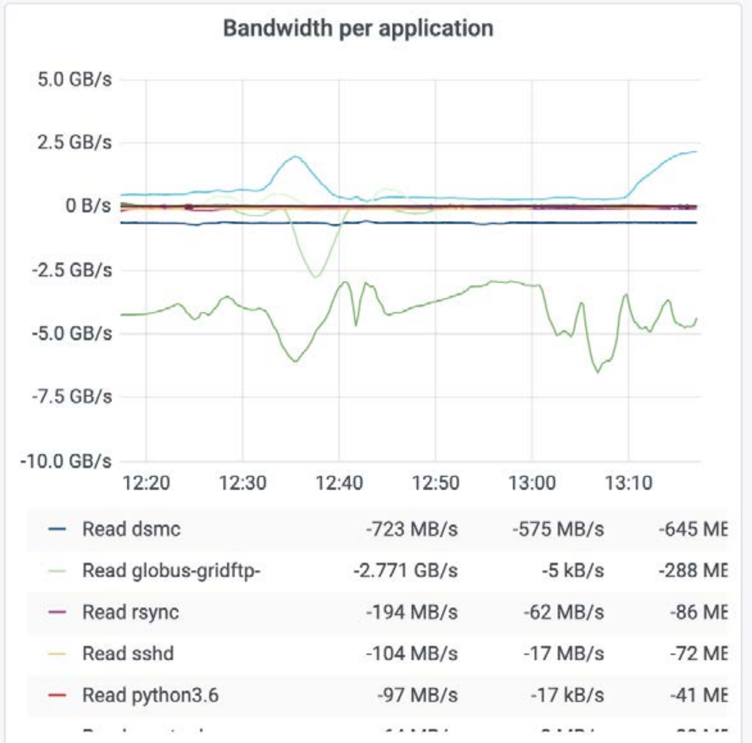


	min	max	avg
Write ev	3 MB/s	823 MB/s	408 MB/s
Write tgi	11 MB/s	154 MB/s	95 MB/s
Write zz:	5 MB/s	219 MB/s	87 MB/s
Write na	5 MB/s	129 MB/s	87 MB/s
Write sa	4 MB/s	83 MB/s	68 MB/s

Adding tags (logins, DTNs ...)

From procname_uid

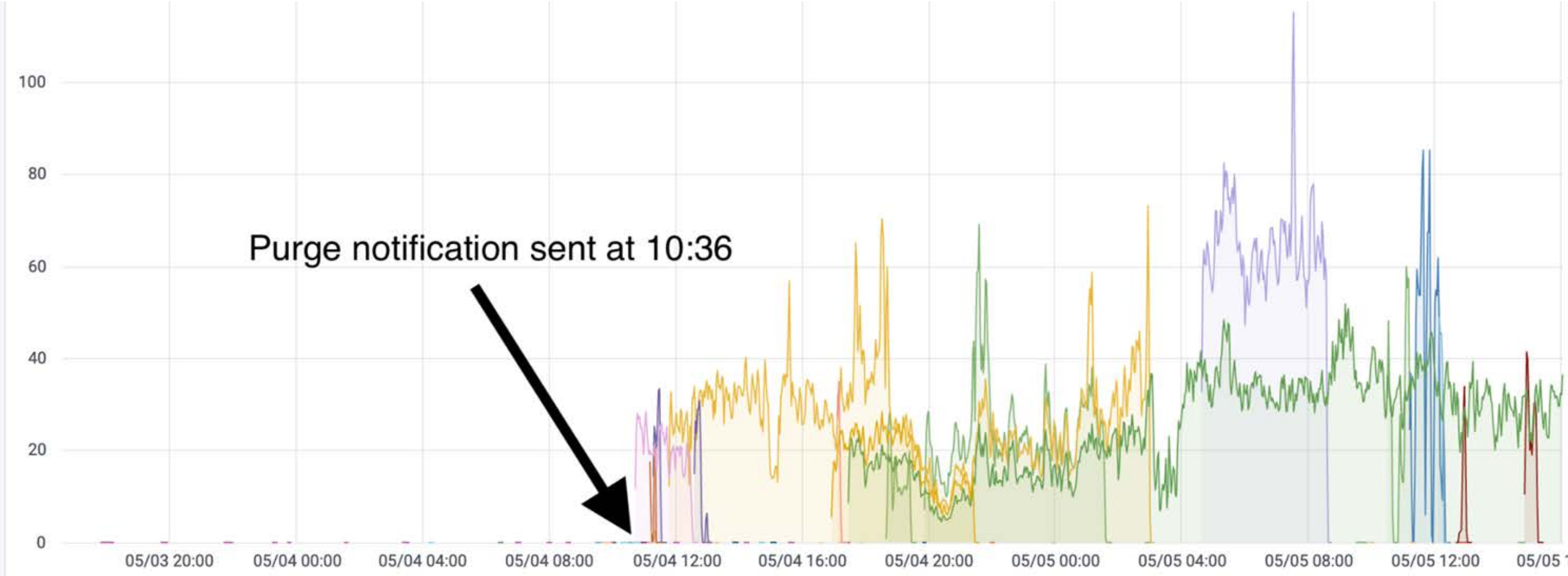
```
lustre_job_write_bytes_total{
component="ost",
jobid="tar.1000",
target="lustre02-OST0000",
fs="lustre02",
application="tar",
user="user1"}
```



—	Read dsmc	-723 MB/s	-575 MB/s	-645 M
—	Read globus-gridftp-	-2.771 GB/s	-5 kB/s	-288 M
—	Read rsync	-194 MB/s	-62 MB/s	-86 M
—	Read sshd	-104 MB/s	-17 MB/s	-72 M
—	Read python3.6	-97 MB/s	-17 kB/s	-41 M

Pattern on login nodes

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

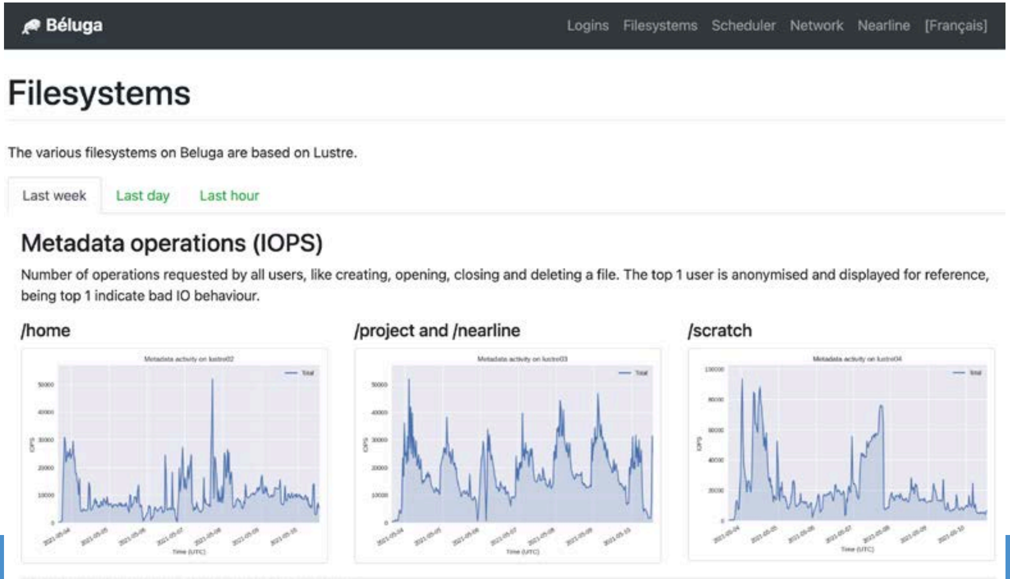


Grafana

- Detailed information is now available to sysadmins
    - Analysts and users does 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



Filesystems

The various filesystems on Beluga are based on Lustre.

Metadata operations (IOPS)

Number of operations requested by all users, like creating, opening, closing and deleting a file. The top 1 user is anonymised and displayed for reference, being top 1 indicate bad IO behaviour.

/home

/project and /nearline

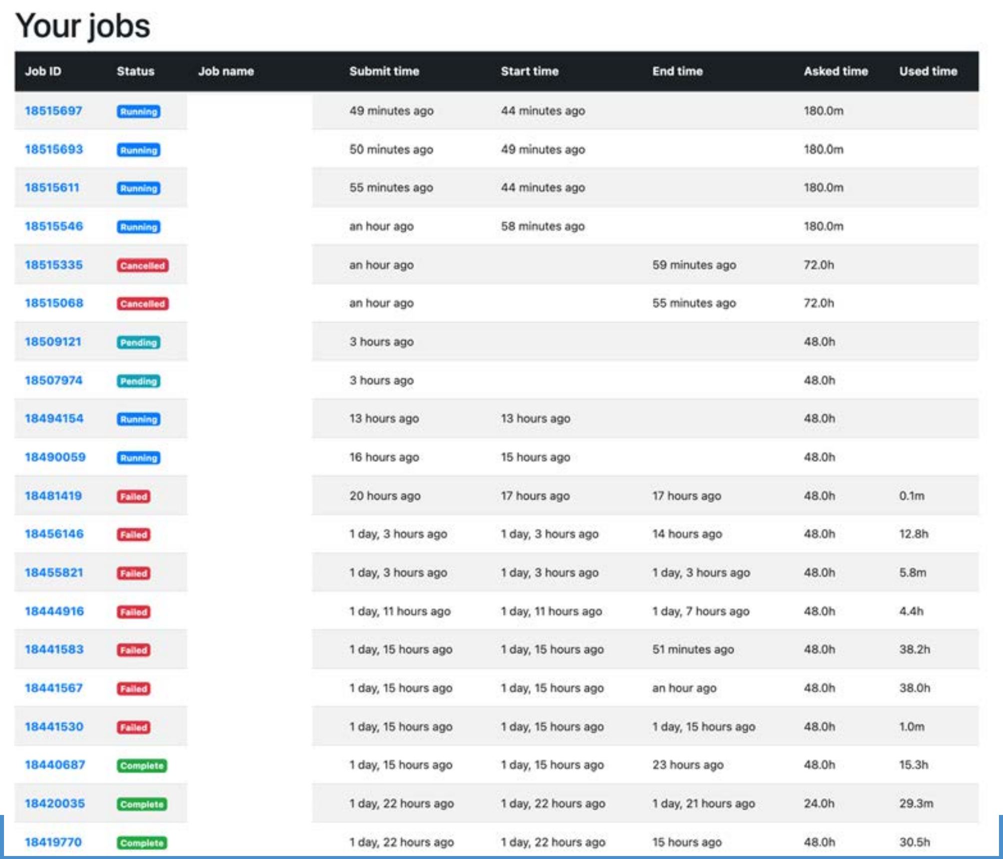
/scratch

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 Shibboleths authentication
  - Translation framework
  - “The web framework for perfectionists with deadlines.”

Recent jobs

- Breakdown per job
- Live job stats

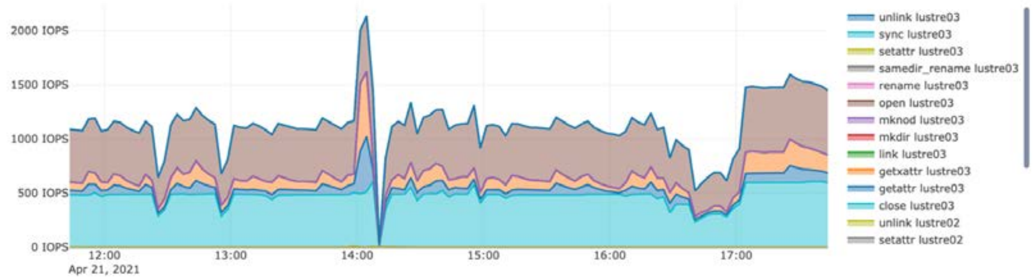


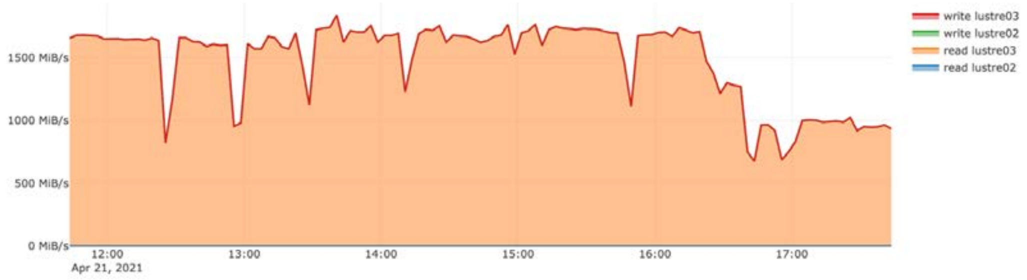
Your jobs

Job ID	Status	Job name	Submit time	Start time	End time	Asked time	Used time
18515697	Running		49 minutes ago	44 minutes ago		180.0m	
18515693	Running		50 minutes ago	49 minutes ago		180.0m	
18515611	Running		55 minutes ago	44 minutes ago		180.0m	
18515546	Running		an hour ago	58 minutes ago		180.0m	
18515335	Cancelled		an hour ago		59 minutes ago	72.0h	
18515068	Cancelled		an hour ago		55 minutes ago	72.0h	
18509121	Pending		3 hours ago			48.0h	
18507974	Pending		3 hours ago			48.0h	
18494154	Running		13 hours ago	13 hours ago		48.0h	
18490059	Running		16 hours ago	15 hours ago		48.0h	
18481419	Failed		20 hours ago	17 hours ago	17 hours ago	48.0h	0.1m
18456146	Failed		1 day, 3 hours ago	1 day, 3 hours ago	14 hours ago	48.0h	12.8h
18455821	Failed		1 day, 3 hours ago	1 day, 3 hours ago	1 day, 3 hours ago	48.0h	5.8m
18444916	Failed		1 day, 11 hours ago	1 day, 11 hours ago	1 day, 7 hours ago	48.0h	4.4h
18441583	Failed		1 day, 15 hours ago	1 day, 15 hours ago	51 minutes ago	48.0h	38.2h
18441567	Failed		1 day, 15 hours ago	1 day, 15 hours ago	an hour ago	48.0h	38.0h
18441530	Failed		1 day, 15 hours ago	1 day, 15 hours ago	1 day, 15 hours ago	48.0h	1.0m
18440687	Complete		1 day, 15 hours ago	1 day, 15 hours ago	23 hours ago	48.0h	15.3h
18420035	Complete		1 day, 22 hours ago	1 day, 22 hours ago	1 day, 21 hours ago	24.0h	29.3m
18419770	Complete		1 day, 22 hours ago	1 day, 22 hours ago	15 hours ago	48.0h	30.5h

Filesystem performance

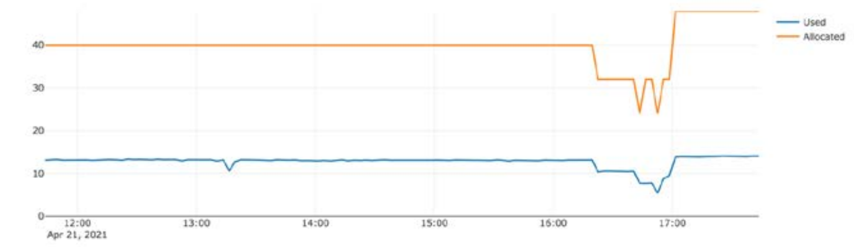
A user can see their own use



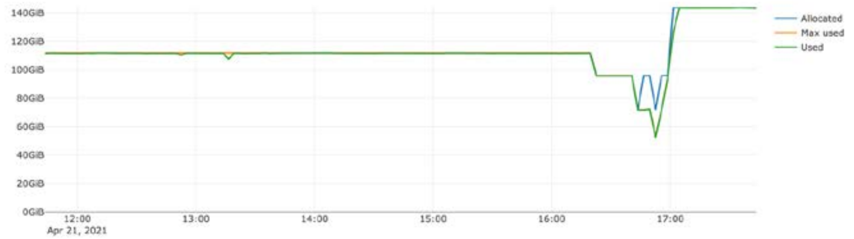


CPU, memory and GPU

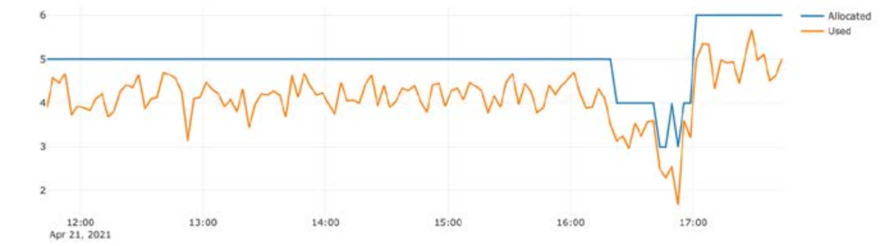
CPU cores

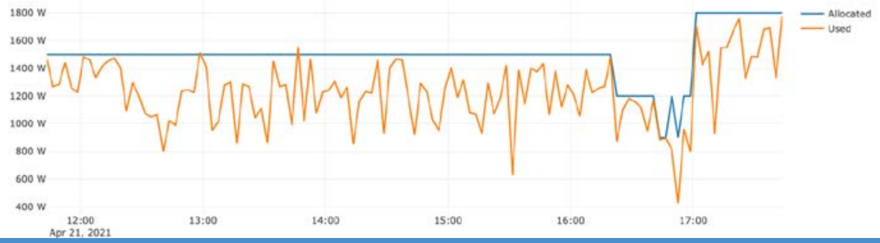


Memory



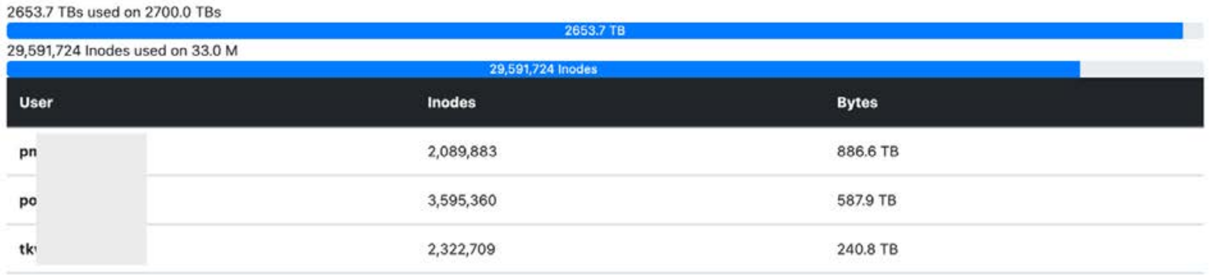
GPUs





Quotas and HSM states

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



2653.7 TBs used on 2700.0 TBs

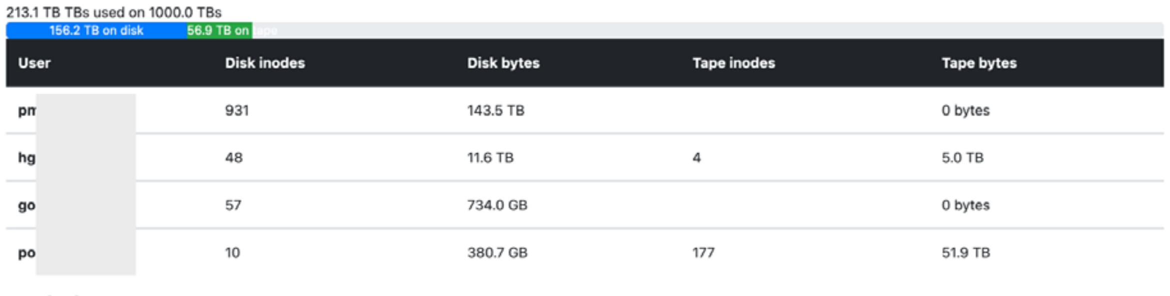
2653.7 TB

29,591,724 Inodes used on 33.0 M

29,591,724 Inodes

User	Inodes	Bytes
pn[redacted]	2,089,883	886.6 TB
po[redacted]	3,595,360	587.9 TB
tk[redacted]	2,322,709	240.8 TB

- HSM (tape) status



213.1 TB TBs used on 1000.0 TBs

156.2 TB on disk 56.9 TB on tape

User	Disk inodes	Disk bytes	Tape inodes	Tape bytes
pn[redacted]	931	143.5 TB		0 bytes
hg[redacted]	48	11.6 TB	4	5.0 TB
go[redacted]	57	734.0 GB		0 bytes
po[redacted]	10	380.7 GB	177	51.9 TB

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

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 a optimized database, resulting in a fast user interface. No wait times once the index is complete.”

User point of view with duc

Calcul Québec

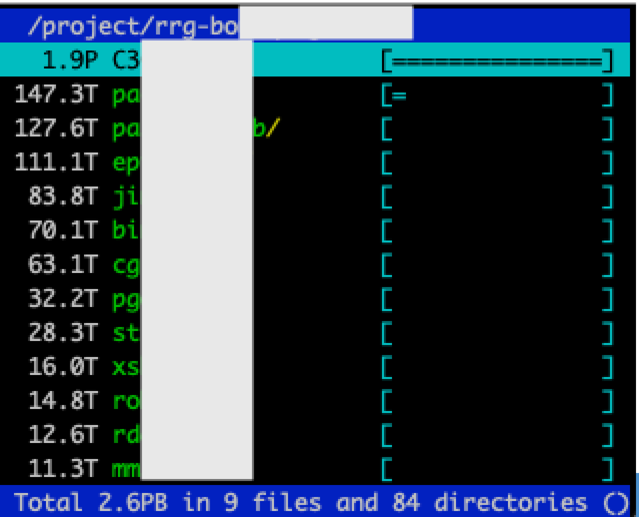
```
$ diskusage_report
[... lfs quota output are here ...]
Disk usage can be explored using the following commands:
diskusage_explorer /home/sigui4          (Last update: 2021-04-21 13:17:48)
diskusage_explorer /scratch/sigui4       (Last update: 2021-04-21 13:20:06)
diskusage_explorer /project/def-sigui4   (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

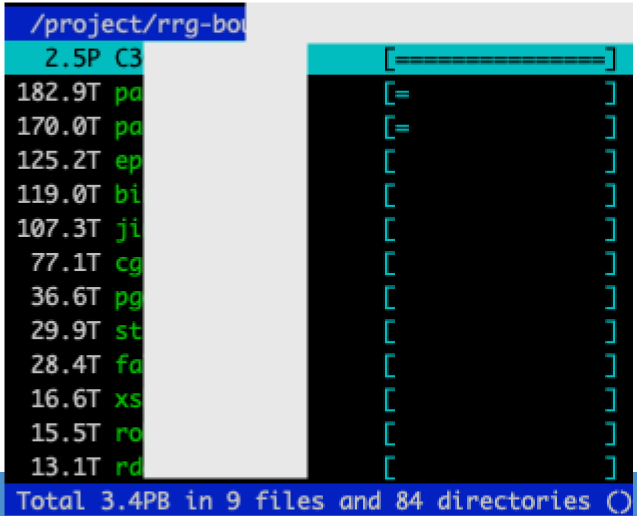
- Can display by actual size, apparent size and inodes
  - compression, HSM and sparse file -> actual != apparent
  - Switching directory take a few ms

Actual



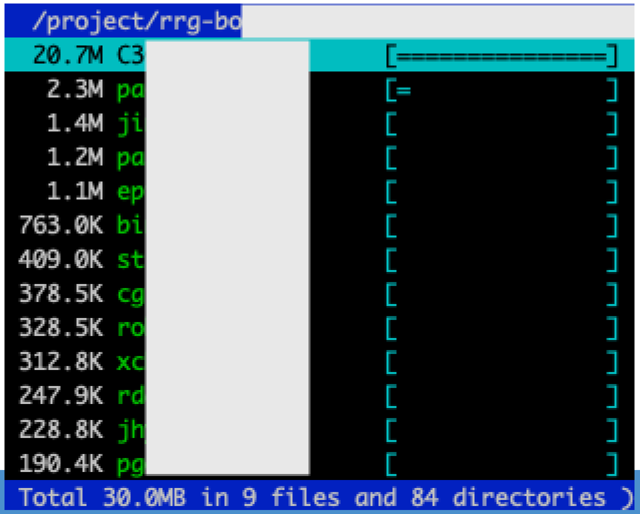
```
 /project/rrg-bo[redacted]
  1.9P C3[redacted]              [===============]
147.3T pa                        [=              ]
127.6T pa        b/              [               ]
111.1T ep                        [               ]
 83.8T ji                        [               ]
 70.1T bi                        [               ]
 63.1T cg                        [               ]
 32.2T pg                        [               ]
 28.3T st                        [               ]
 16.0T xs                        [               ]
 14.8T ro                        [               ]
 12.6T rd                        [               ]
 11.3T mm                        [               ]
Total 2.6PB in 9 files and 84 directories ()
```

Apparent



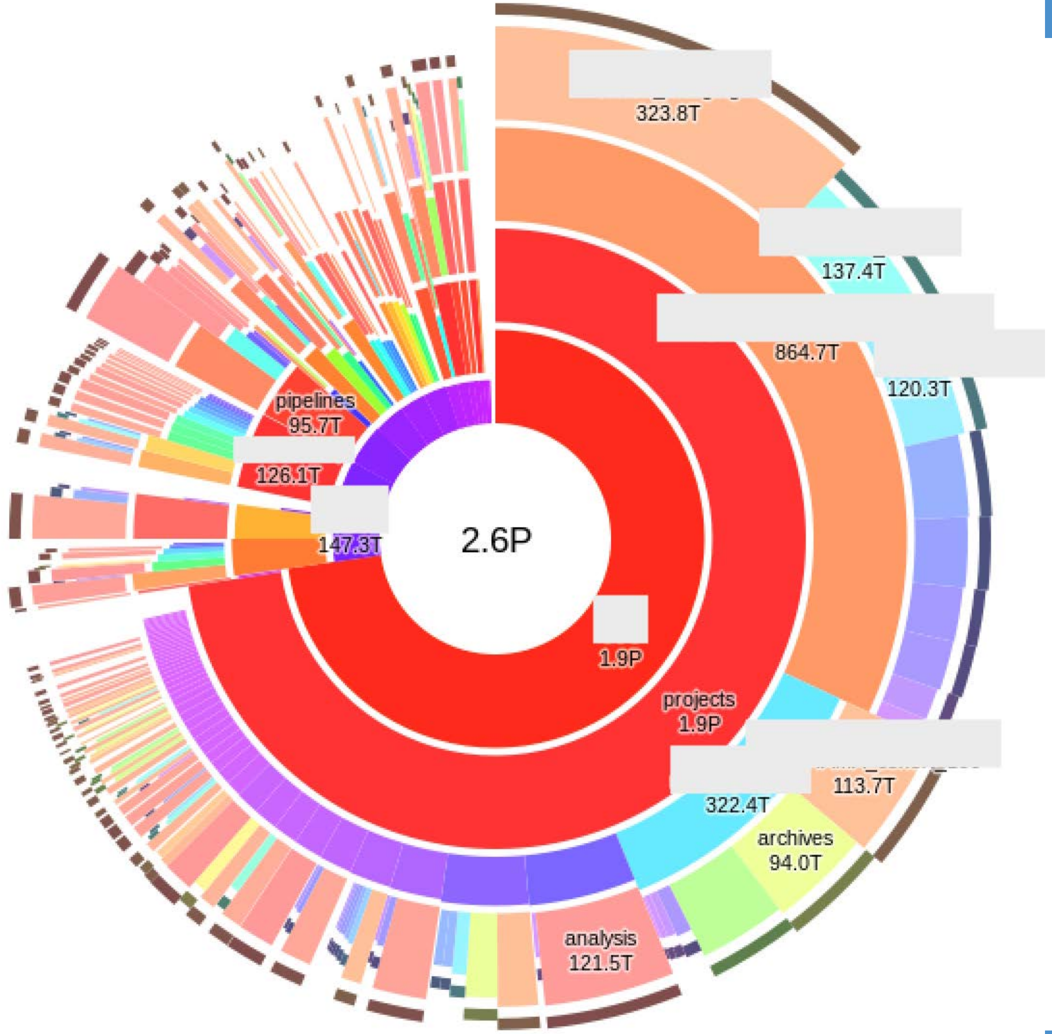
```
 /project/rrg-bo[redacted]
  2.5P C3[redacted]              [===============]
182.9T pa                        [=              ]
170.0T pa                        [=              ]
125.2T ep                        [               ]
119.0T bi                        [               ]
107.3T ji                        [               ]
 77.1T cg                        [               ]
 36.6T pg                        [               ]
 29.9T st                        [               ]
 28.4T fa                        [               ]
 16.6T xs                        [               ]
 15.5T ro                        [               ]
 13.1T rd                        [               ]
Total 3.4PB in 9 files and 84 directories ()
```

Inodes



```
 /project/rrg-bo[redacted]
 20.7M C3[redacted]              [===============]
  2.3M pa                        [=              ]
  1.4M ji                        [               ]
  1.2M pa                        [               ]
  1.1M ep                        [               ]
763.0K bi                        [               ]
409.0K st                        [               ]
378.5K cg                        [               ]
328.5K ro                        [               ]
312.8K xc                        [               ]
247.9K rd                        [               ]
228.8K jh                        [               ]
190.4K pg                        [               ]
Total 30.0MB in 9 files and 84 directories )
```

X GUI



Calcul Québec

duc

- 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/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

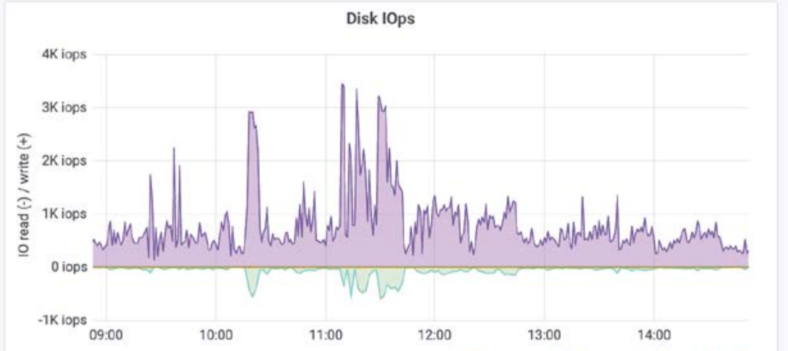
Calcul Québec

- 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...

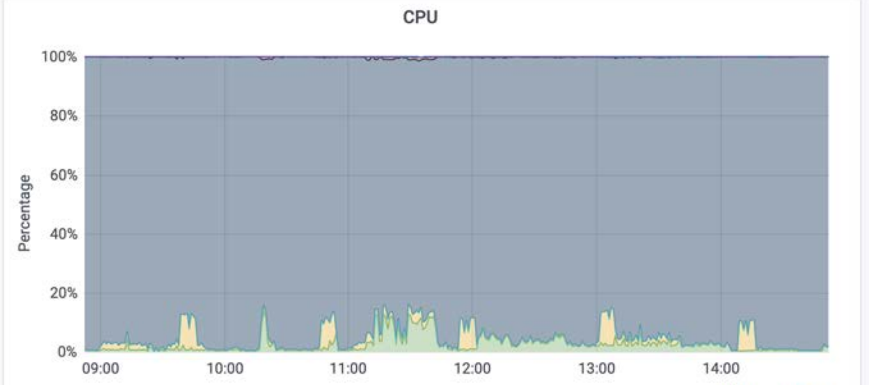
robinhood2duc

- 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

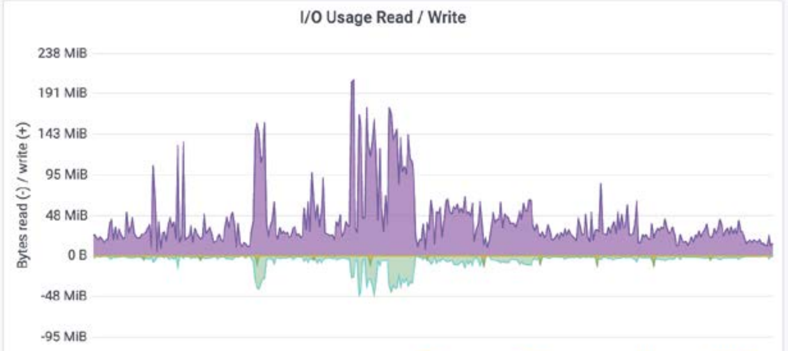
DB server metrics



	min	max	avg	current
— md0 - Reads completed	0 iops	18 iops	1 iops	0 iops
— nvme0n1 - Reads completed	3 iops	611 iops	78 iops	16 iops
— nvme1n1 - Reads completed	3 iops	589 iops	78 iops	15 iops
— sda - Reads completed	0 iops	17 iops	1 iops	0 iops
— sdb - Reads completed	0 iops	1 iops	0 iops	0 iops



	min	max
— System - Processes executing in kernel mode	11.30	1.10 K
— User - Normal processes executing in user mode	5.03	1.02 K
— Nice - Niced processes executing in user mode	0	0
— Idle - Waiting for something to happen	6.57 K	7.98 K
— Iowait - Waiting for I/O to complete	1.60	103.16



	min	max	avg	current
— md0 - Successfully read bytes	0 B	12.939 MiB	308.926 KiB	1.067 KiB
— nvme0n1 - Successfully read bytes	231.967 KiB	48.095 MiB	5.677 MiB	1.114 MiB
— nvme1n1 - Successfully read bytes	218.550 KiB	46.949 MiB	5.659 MiB	1.005 MiB
— sda - Successfully read bytes	0 B	9.736 MiB	237.874 KiB	1.067 KiB
— sdb - Successfully read bytes	0 B	3.258 MiB	71.033 KiB	0 B

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