Diagnosing Wasted Resources from User Facing Portals on the National Clusters

Compute Ontario Colloquium: SHARCNET

Tyler Collins
HPC Analyst, Brock University

Outline and format for today's talk

- Summarize previous talks and assumptions
- Discuss wait times
  - What's fair vs what a user needs
- How to investigate an account's priority that is seeing wait times
- **Diagnosing wasted resources**
  - CLI tools
  - Portals

Open question period at the end of the talk!

Send tickets to help@sharcnet.ca

Previous talks and assumptions

All talks by James Desjardins:

- Visualizing job properties for wait time assessment
- Exploring job wait times on Alliance compute clusters: a holistic view
- Exploring Compute Usage from User Facing Portals on the National Clusters

As this is the latest in a series, we'll be building off of these talks

Safe to conclude that wait times can be unpredictable for users

Previous talks and assumptions: definitions

- **Job shape:** phrase meaning job attributes taken as a whole
  - 16 cores x 4G x 7 days
- **Billing:** combined metric that the scheduler uses to judge how many resources are used
  - A single core 125G job is not billed as a single core, but instead an entire node
- **Waste:** resources that have been requested but NOT used
  - Requesting a GPU when your software does not use an accelerator
- **Allocation:** where usage of the system is “billed” too, often named after a PI
  - Default allocations start with def and have an average target
  - RAC accounts are competition based assurances of targets
- **Priority:** how the scheduler decides the ordering of which jobs to execute next
  - You CANNOT “bank” priority, but you should still be allowed to catch up

Previous talks and assumptions: summary

Researchers often experience wait times without knowing what is causing them

This can be addressed by:

- Understanding priority and job shapes
  - For example, age in queue does practically nothing
- Better CLI tools like cluster-stats
- Python packages and tools to produces visualizations

The strongest way to minimize wait times is to minimize waste and optimize your job shape

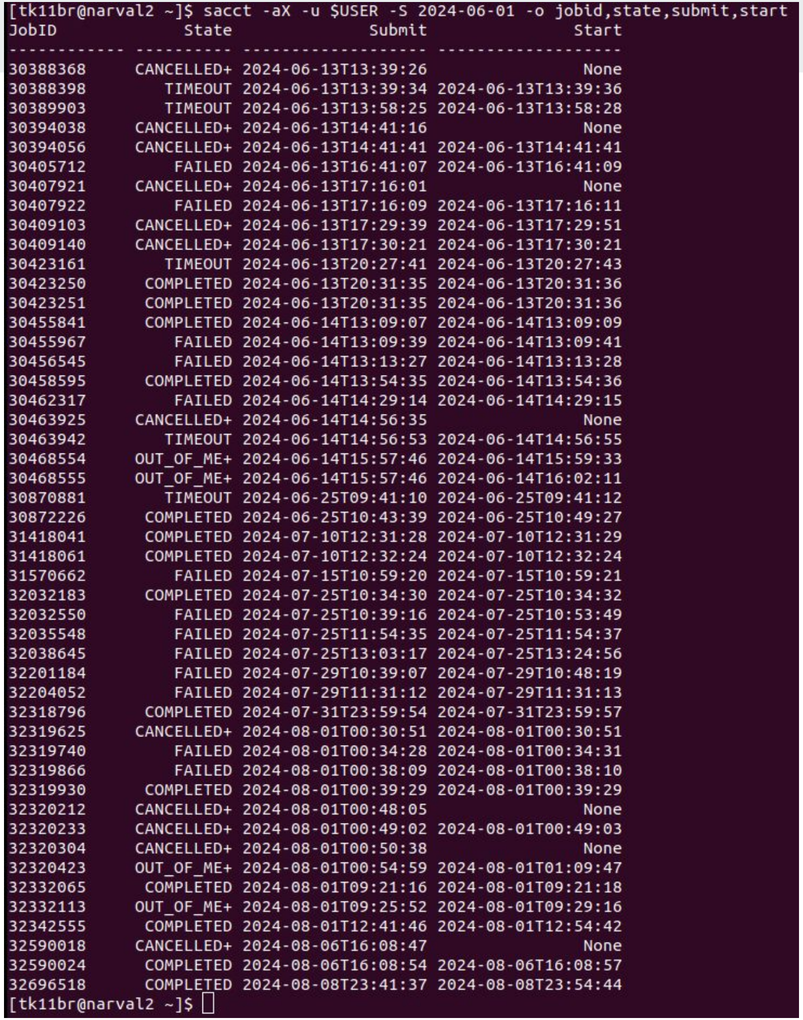
Wait times and where to find them

First most important principle: wait times are NORMAL in a fair system

The question is: what is an acceptable wait time?

- Depends entirely on your field of study
- If you need a heuristic: if you can rerun your entire thesis in less than a week you likely don’t need to go further

As before: “The strongest way to minimize wait times is to minimize waste and optimize your job shape”



```
[tk11br@narval2 ~]$ sacct -aX -u $USER -S 2024-06-01 -o jobid,state,submit,start
JobID             State              Submit               Start
------------ ---------- ------------------- -------------------
30388368     CANCELLED+ 2024-06-13T13:39:26                None
30388398        TIMEOUT 2024-06-13T13:39:34 2024-06-13T13:39:36
30389903        TIMEOUT 2024-06-13T13:58:25 2024-06-13T13:58:28
30394038     CANCELLED+ 2024-06-13T14:41:16                None
30394056     CANCELLED+ 2024-06-13T14:41:41 2024-06-13T14:41:41
30405712         FAILED 2024-06-13T16:41:07 2024-06-13T16:41:09
30407921     CANCELLED+ 2024-06-13T17:16:01                None
30407922         FAILED 2024-06-13T17:16:09 2024-06-13T17:16:11
30409103     CANCELLED+ 2024-06-13T17:29:39 2024-06-13T17:29:51
30409140     CANCELLED+ 2024-06-13T17:30:21 2024-06-13T17:30:21
30423161        TIMEOUT 2024-06-13T20:27:41 2024-06-13T20:27:43
30423250      COMPLETED 2024-06-13T20:31:35 2024-06-13T20:31:36
30423251      COMPLETED 2024-06-13T20:31:35 2024-06-13T20:31:36
30455841      COMPLETED 2024-06-14T13:09:07 2024-06-14T13:09:09
30455967         FAILED 2024-06-14T13:09:39 2024-06-14T13:09:41
30456545         FAILED 2024-06-14T13:13:27 2024-06-14T13:13:28
30458595      COMPLETED 2024-06-14T13:54:35 2024-06-14T13:54:36
30462317         FAILED 2024-06-14T14:29:14 2024-06-14T14:29:15
30463925     CANCELLED+ 2024-06-14T14:56:35                None
30463942        TIMEOUT 2024-06-14T14:56:53 2024-06-14T14:56:55
30468554     OUT_OF_ME+ 2024-06-14T15:57:46 2024-06-14T15:59:33
30468555     OUT_OF_ME+ 2024-06-14T15:57:46 2024-06-14T16:02:11
30870881        TIMEOUT 2024-06-25T09:41:10 2024-06-25T09:41:12
30872226      COMPLETED 2024-06-25T10:43:39 2024-06-25T10:49:27
31418041      COMPLETED 2024-07-10T12:31:28 2024-07-10T12:31:29
31418061      COMPLETED 2024-07-10T12:32:24 2024-07-10T12:32:24
31570662         FAILED 2024-07-15T10:59:20 2024-07-15T10:59:21
32032183      COMPLETED 2024-07-25T10:34:30 2024-07-25T10:34:32
32032550         FAILED 2024-07-25T10:39:16 2024-07-25T10:53:49
32035548         FAILED 2024-07-25T11:54:35 2024-07-25T11:54:37
32038645         FAILED 2024-07-25T13:03:17 2024-07-25T13:24:56
32201184         FAILED 2024-07-29T10:39:07 2024-07-29T10:48:19
32204052         FAILED 2024-07-29T11:31:12 2024-07-29T11:31:13
32318796      COMPLETED 2024-07-31T23:59:54 2024-07-31T23:59:57
32319625     CANCELLED+ 2024-08-01T00:30:51 2024-08-01T00:30:51
32319740         FAILED 2024-08-01T00:34:28 2024-08-01T00:34:31
32319866         FAILED 2024-08-01T00:38:09 2024-08-01T00:38:10
32319930      COMPLETED 2024-08-01T00:39:29 2024-08-01T00:39:29
32320212     CANCELLED+ 2024-08-01T00:48:05                None
32320233     CANCELLED+ 2024-08-01T00:49:02 2024-08-01T00:49:03
32320304     CANCELLED+ 2024-08-01T00:50:38                None
32320423     OUT_OF_ME+ 2024-08-01T00:54:59 2024-08-01T01:09:47
32332065      COMPLETED 2024-08-01T09:21:16 2024-08-01T09:21:18
32332113     OUT_OF_ME+ 2024-08-01T09:25:52 2024-08-01T09:29:16
32342555      COMPLETED 2024-08-01T12:41:46 2024-08-01T12:54:42
32590018     CANCELLED+ 2024-08-06T16:08:47                None
32590024      COMPLETED 2024-08-06T16:08:54 2024-08-06T16:08:57
32696518      COMPLETED 2024-08-08T23:41:37 2024-08-08T23:54:44
[tk11br@narval2 ~]$
```

Wait times and where to find them: my account

Some quick ways to see your jobs and job history:

- “squeue -u $USER”
- “sacct -aX -u $USER -s 2024-06-01 -o jobid,state,submit,start”
  - Write output to file and pass to excel
  - View our previous talks for Python packages that can calculate these summaries

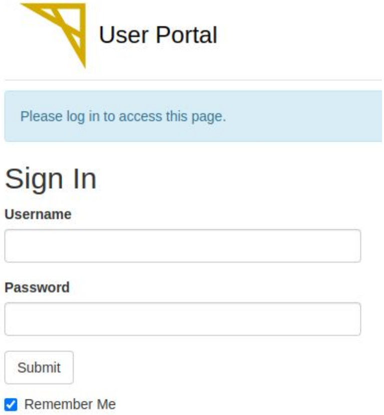
If you’re with me so far, let’s begin diagnosing that waste!

Let’s pretend I have been sent a ticket that seeks to minimize wait times

But WHY am I waiting?

First step is to simply check the priority over time of the account you are submitting to

This is best done via the portal developed by Sergiy Khan located at: portal.alliancecan.ca



User Portal

Please log in to access this page.

Sign In

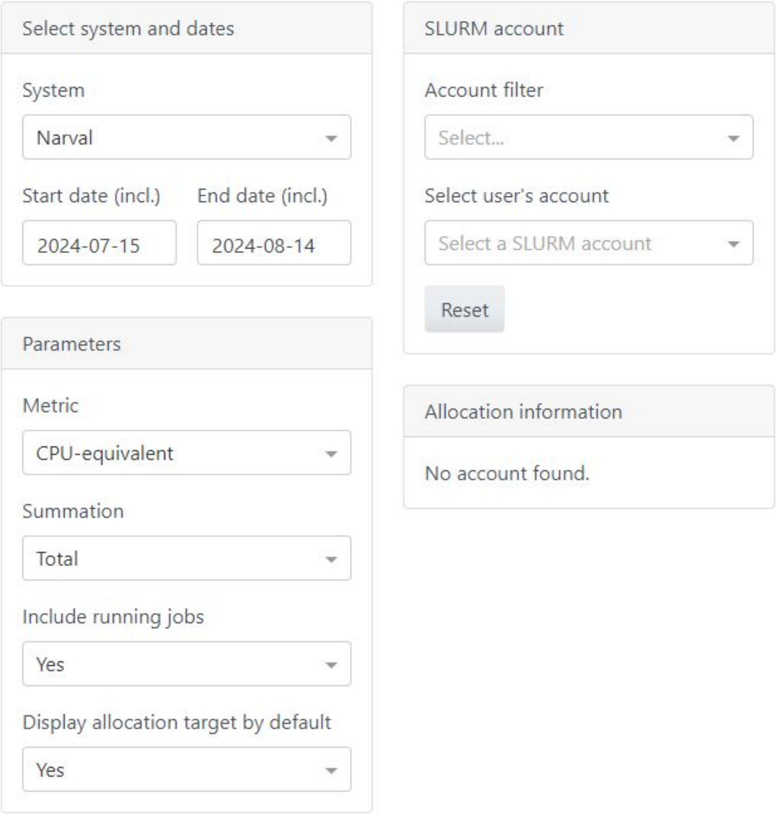
Username

Password

☑ Remember Me

Portal Options

- SLURM Account fields change which group you are investigating
- System and dates are self explanatory
- Parameters box allow for exploring CPU/GPU billing as well as view cumulative usage
- Allocation information will be pulled from our databases and populate with the correct targets



Select system and dates

System

Narval

Start date (incl.) End date (incl.)

2024-07-15 2024-08-14

Parameters

Metric

CPU-equivalent

Summation

Total

Include running jobs

Yes

Display allocation target by default

Yes

SLURM account

Account filter

Select...

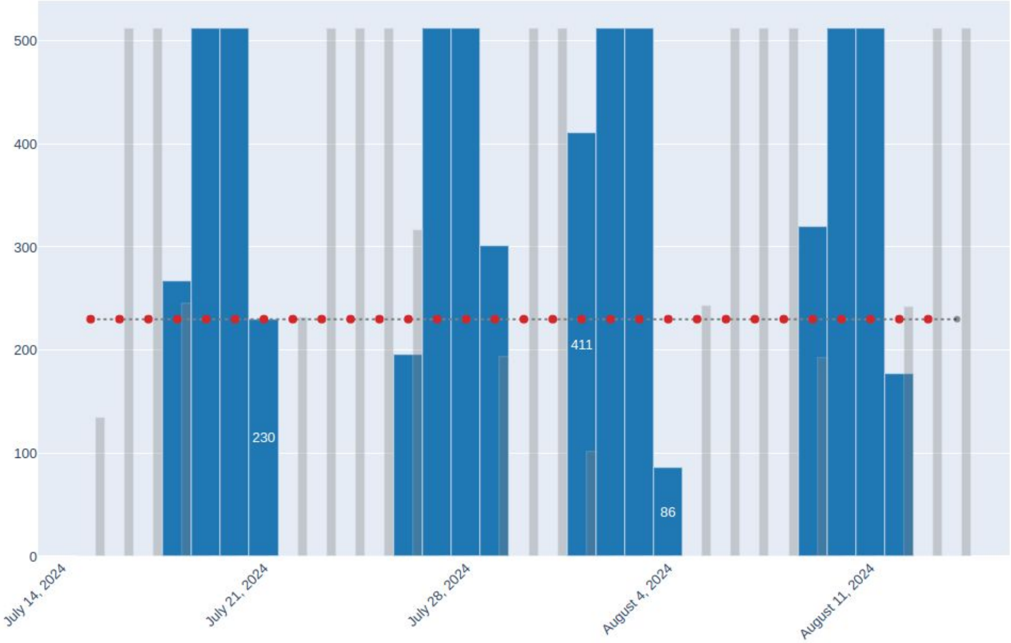
Select user's account

Select a SLURM account

Allocation information

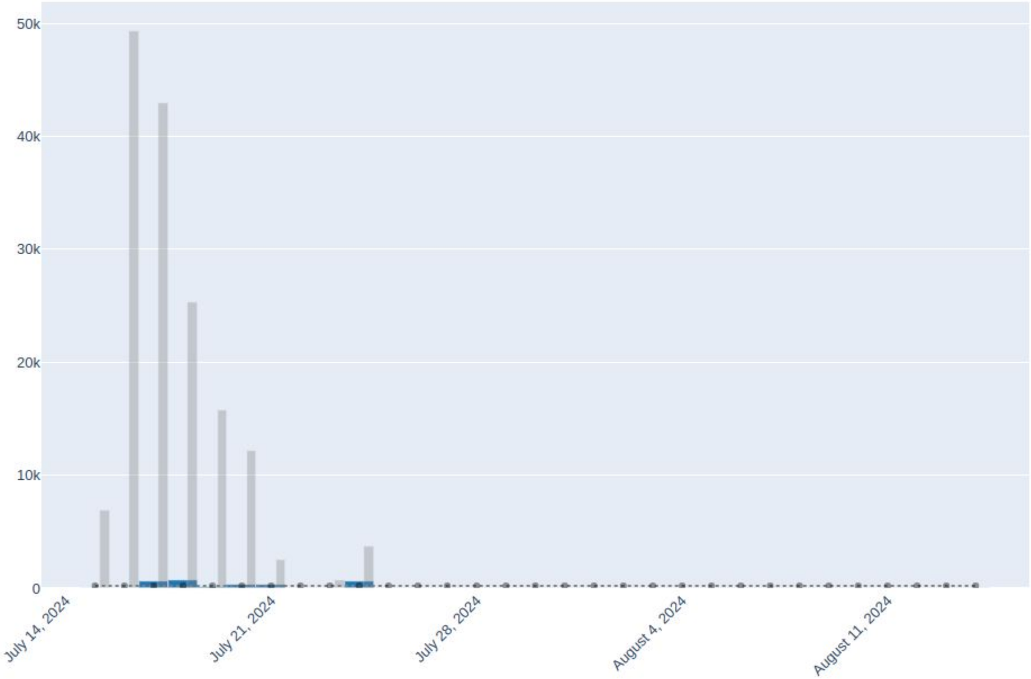
No account found.

A regular healthy account

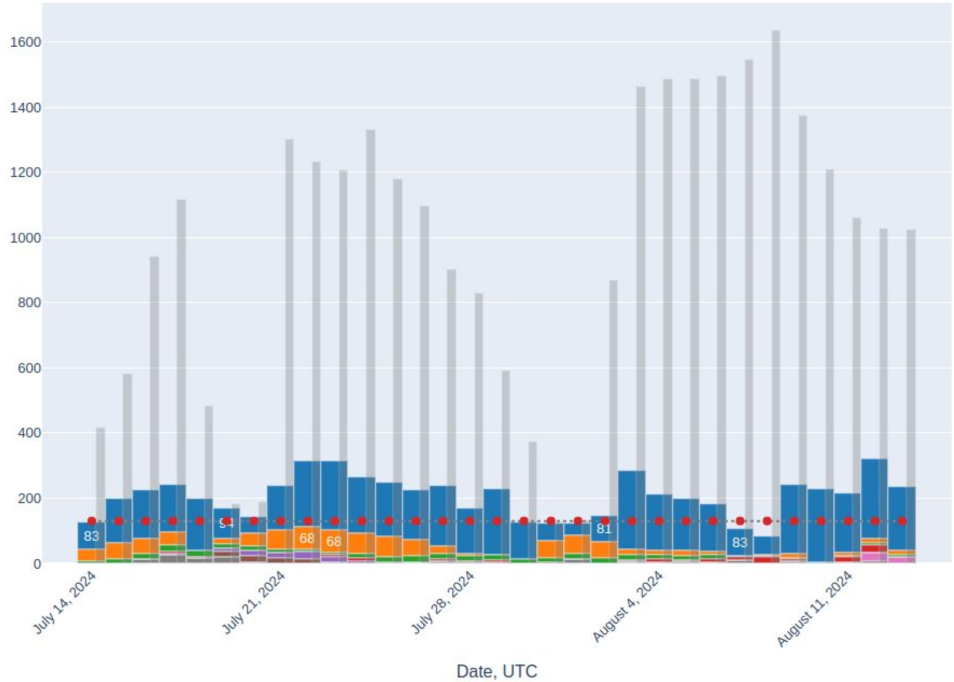


SHARCNET: Tyler Collins

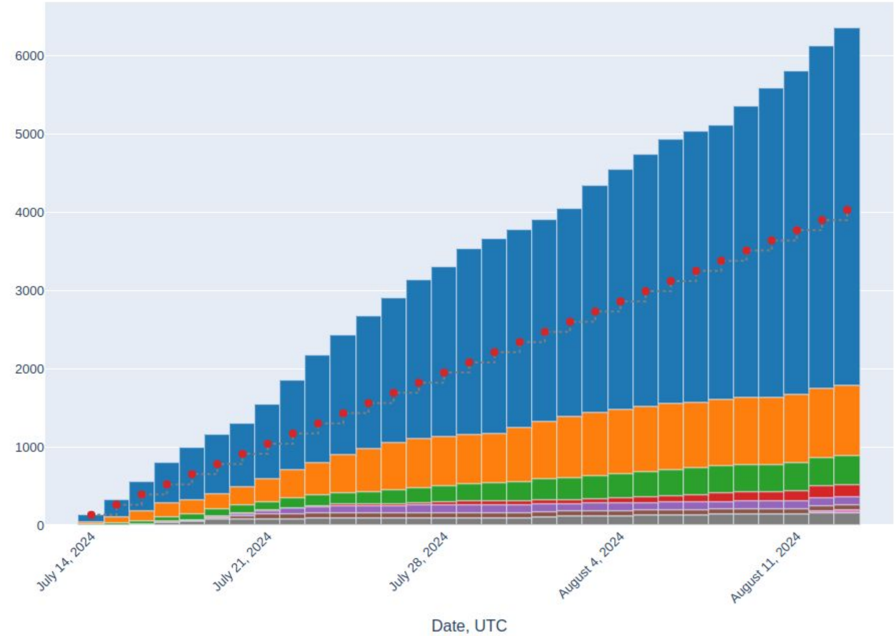
An underserved account



A large overserved account



A large overserved account: cumulative



Priority problems

Two main cases, and a third hidden corner case:

1. The account is not at its allocation target and is experiencing wait times
    a. 99.9999% of the time this is fixable with better job shape parameters
2. The account is at or over its allocation target and is experiencing wait times
    a. 80% of the time this is fixable with better job shape parameters
    b. The other 20% is a good place to start a ticket with us
3. You are competing against other smaller jobs WITHIN your account
    a. Chat with your supervisor or explore using another system
    b. Can still consider optimizing job shape to make your own jobs more favorable

Recalling previous talks

Eliminating waste is your best bet in all cases

Consider that a job that can run anywhere with less specifications has more chances to “jump the queue”

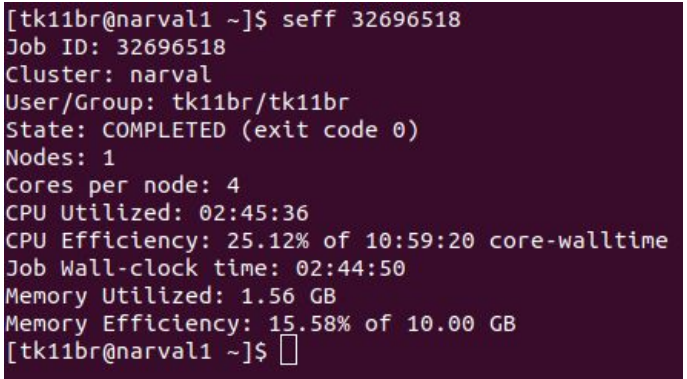
These questions often make really good tickets, don’t hesitate to reach out

Let’s move on to metrics that actually represent waste

Eliminating waste: low hanging fruit

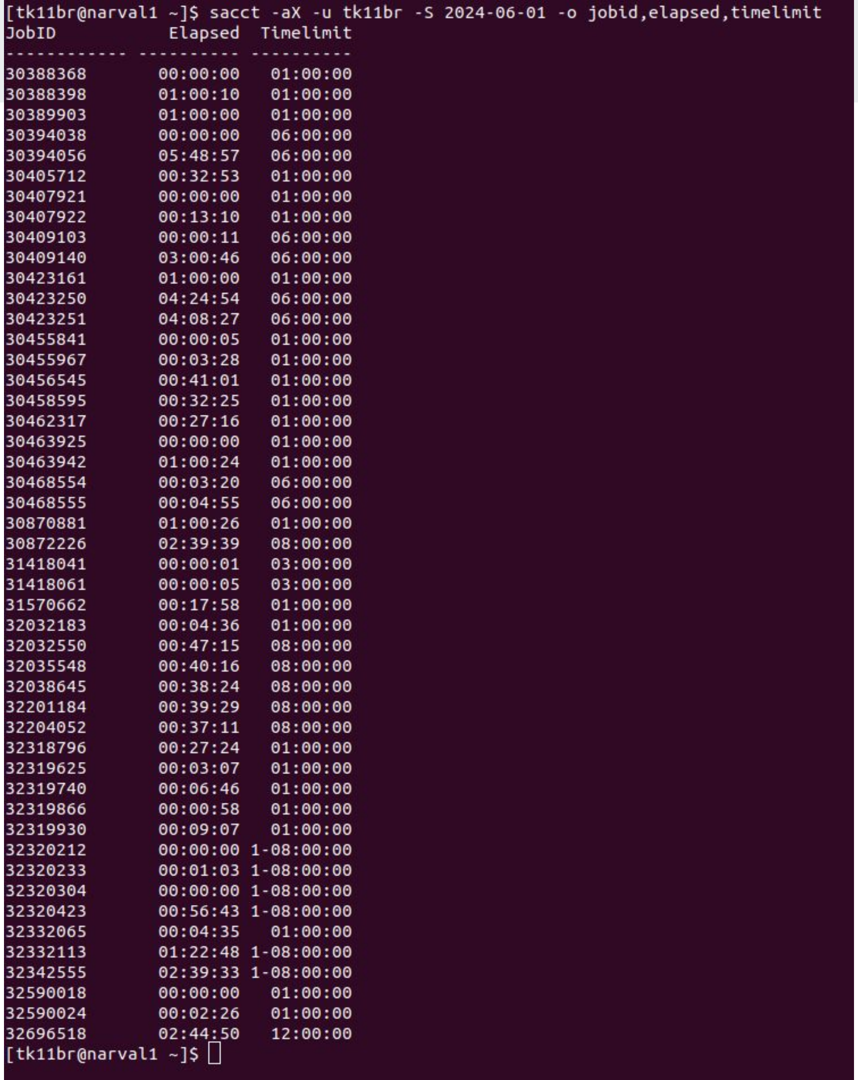
- Actually look at the run times and provide a more accurate estimate
  - If you know that everything runs in 3-hours 5% of the time, submit them all, and resubmit the ones that fail with a slightly longer interval, OR better yet, checkpointing
- Use "seff" to look at your memory and core efficiency
  - Seff isn't always accurate and can be confused by multi-node jobs, strange memory allocation patterns, etc

What if it's a complex job?



```
[tk11br@narval1 ~]$ seff 32696518
Job ID: 32696518
Cluster: narval
User/Group: tk11br/tk11br
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 4
CPU Utilized: 02:45:36
CPU Efficiency: 25.12% of 10:59:20 core-walltime
Job Wall-clock time: 02:44:50
Memory Utilized: 1.56 GB
Memory Efficiency: 15.58% of 10.00 GB
[tk11br@narval1 ~]$ 
```

Oops...



```
[tk11br@narval1 ~]$ sacct -aX -u tk11br -S 2024-06-01 -o jobid,elapsed,timelimit
JobID           Elapsed  Timelimit
------------ ---------- ----------
30388368       00:00:00   01:00:00
30388398       01:00:10   01:00:00
30389903       01:00:00   01:00:00
30394038       00:00:00   06:00:00
30394056       05:48:57   06:00:00
30405712       00:32:53   01:00:00
30407921       00:00:00   01:00:00
30407922       00:13:10   01:00:00
30409103       00:00:11   06:00:00
30409140       03:00:46   06:00:00
30423161       01:00:00   01:00:00
30423250       04:24:54   06:00:00
30423251       04:08:27   06:00:00
30455841       00:00:05   01:00:00
30455967       00:03:28   01:00:00
30456545       00:41:01   01:00:00
30458595       00:32:25   01:00:00
30462317       00:27:16   01:00:00
30463925       00:00:00   01:00:00
30463942       01:00:24   01:00:00
30468554       00:03:20   06:00:00
30468555       00:04:55   06:00:00
30870881       01:00:26   01:00:00
30872226       02:39:39   08:00:00
31418041       00:00:01   03:00:00
31418061       00:00:05   03:00:00
31570662       00:17:58   01:00:00
32032183       00:04:36   01:00:00
32032550       00:47:15   08:00:00
32035548       00:40:16   08:00:00
32038645       00:38:24   08:00:00
32201184       00:39:29   08:00:00
32204052       00:37:11   08:00:00
32318796       00:27:24   01:00:00
32319625       00:03:07   01:00:00
32319740       00:06:46   01:00:00
32319866       00:00:58   01:00:00
32319930       00:09:07   01:00:00
32320212       00:00:00 1-08:00:00
32320233       00:01:03 1-08:00:00
32320304       00:00:00 1-08:00:00
32320423       00:56:43 1-08:00:00
32332065       00:04:35   01:00:00
32332113       01:22:48 1-08:00:00
32342555       02:39:33 1-08:00:00
32590018       00:00:00   01:00:00
32590024       00:02:26   01:00:00
32696518       02:44:50   12:00:00
[tk11br@narval1 ~]$ 
```

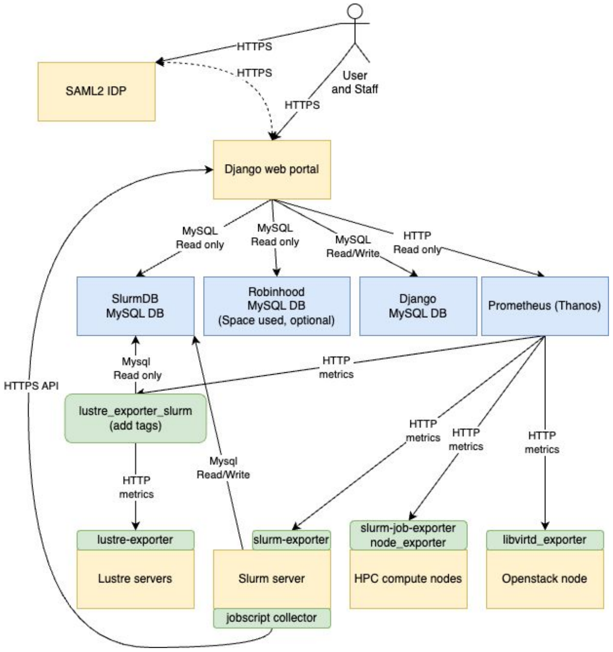
Introducing the userportal

Primarily developed by Simon Guilbault

- Can be found at: https://portail.narval.calculquebec.ca/
- Leverages Prometheus and Thanos to report metrics from different sources
  - Compute nodes, login nodes, filesystem resources, etc
- Very high temporal resolution
  - Do not need to wait for the job to complete
- Most features available on Narval and Beluga, with a development version deployed for Graham
  - Principles learned from these principles will generalize to every system

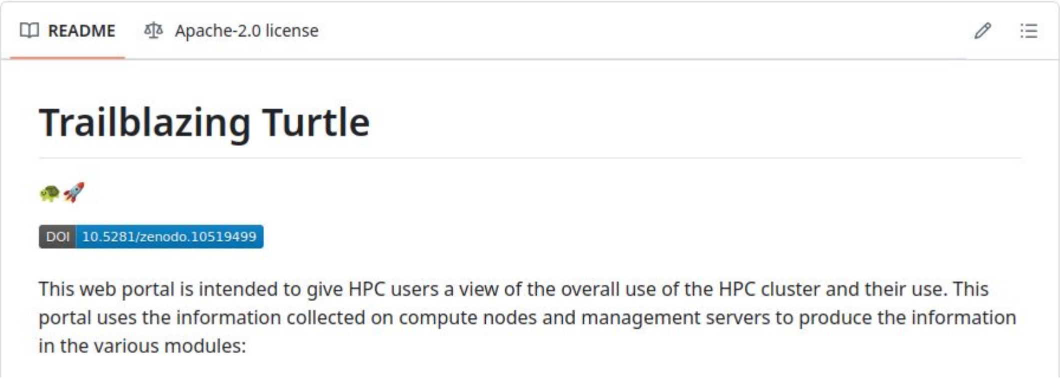
Allow users to see exactly what a job is doing at a low level

Userportal architecture



Codebase for the userportal

Can be found on github here: https://github.com/guilbaults/TrailblazingTurtle



Trailblazing Turtle

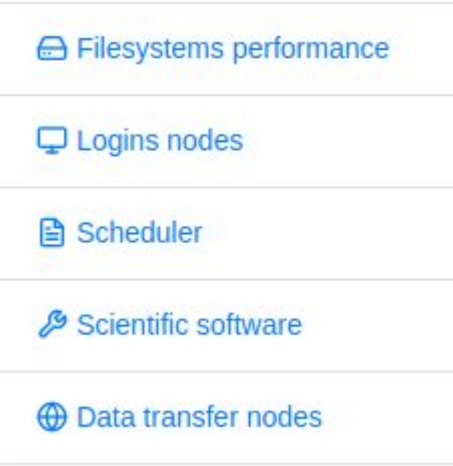
🐢🚀

DOI 10.5281/zenodo.10519499

This web portal is intended to give HPC users a view of the overall use of the HPC cluster and their use. This portal uses the information collected on compute nodes and management servers to produce the information in the various modules:

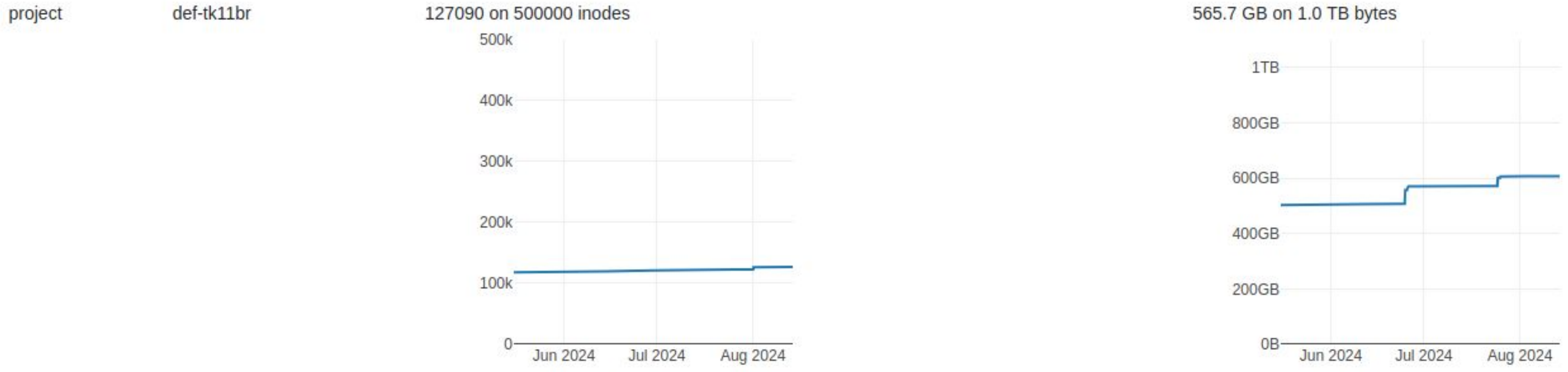
My own personal userportal page

Let’s take a look at my own personal page and explore what can be found on the homepage of the portal



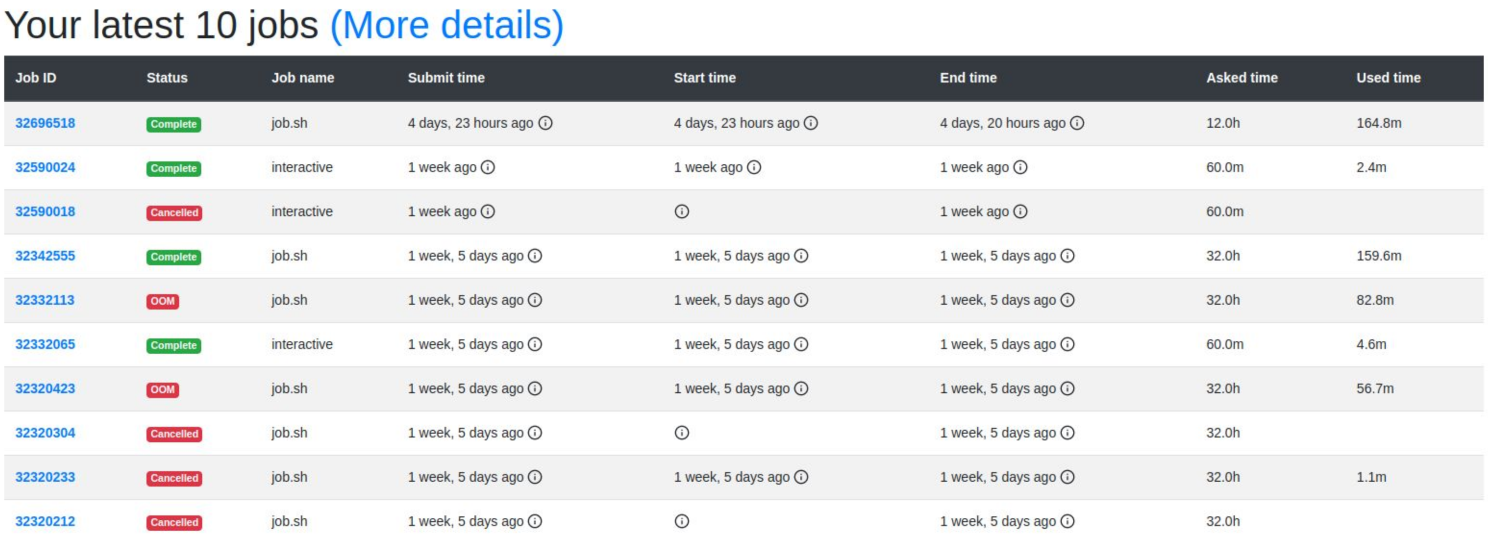
- Filesystems performance
- Logins nodes
- Scheduler
- Scientific software
- Data transfer nodes

Filesystem usage and quotas



project	def-tk11br	127090 on 500000 inodes	565.7 GB on 1.0 TB bytes

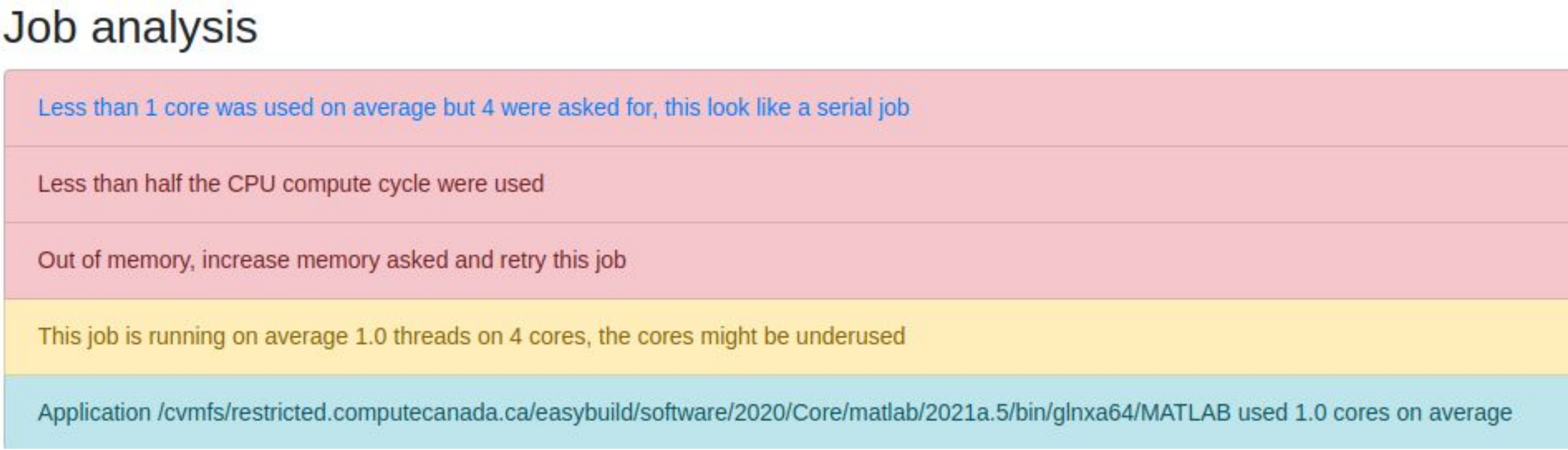
At a glance job summaries



Your latest 10 jobs (More details)

Job ID	Status	Job name	Submit time	Start time	End time	Asked time	Used time
32696518	Complete	job.sh	4 days, 23 hours ago ⓘ	4 days, 23 hours ago ⓘ	4 days, 20 hours ago ⓘ	12.0h	164.8m
32590024	Complete	interactive	1 week ago ⓘ	1 week ago ⓘ	1 week ago ⓘ	60.0m	2.4m
32590018	Cancelled	interactive	1 week ago ⓘ	ⓘ	1 week ago ⓘ	60.0m	
32342555	Complete	job.sh	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	32.0h	159.6m
32332113	OOM	job.sh	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	32.0h	82.8m
32332065	Complete	interactive	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	60.0m	4.6m
32320423	OOM	job.sh	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	32.0h	56.7m
32320304	Cancelled	job.sh	1 week, 5 days ago ⓘ	ⓘ	1 week, 5 days ago ⓘ	32.0h	
32320233	Cancelled	job.sh	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	32.0h	1.1m
32320212	Cancelled	job.sh	1 week, 5 days ago ⓘ	ⓘ	1 week, 5 days ago ⓘ	32.0h	

Picking a single job and exploring



Job analysis

Less than 1 core was used on average but 4 were asked for, this look like a serial job

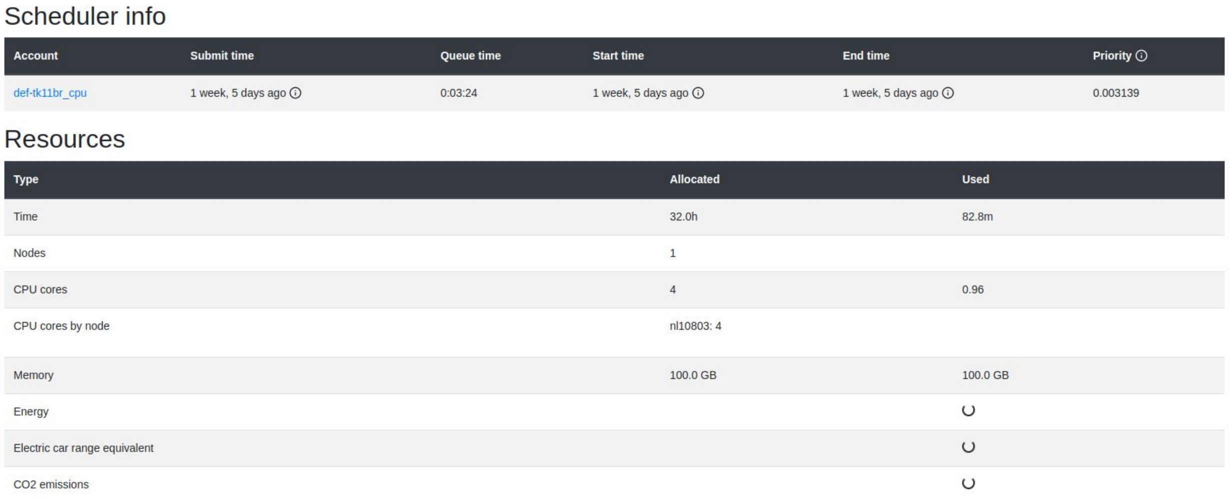
Less than half the CPU compute cycle were used

Out of memory, increase memory asked and retry this job

This job is running on average 1.0 threads on 4 cores, the cores might be underused

Application /cvmfs/restricted.computecanada.ca/easybuild/software/2020/Core/matlab/2021a.5/bin/glnxa64/MATLAB used 1.0 cores on average

Tabular summary information



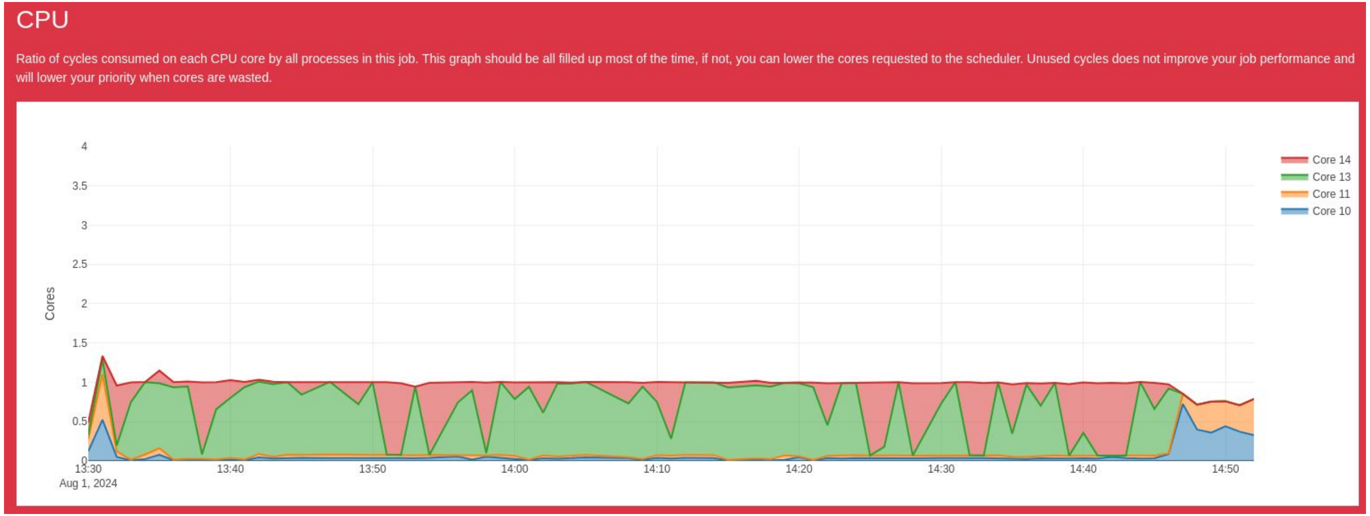
Scheduler info

Account	Submit time	Queue time	Start time	End time	Priority ⓘ
def-tk11br_cpu	1 week, 5 days ago ⓘ	0:03:24	1 week, 5 days ago ⓘ	1 week, 5 days ago ⓘ	0.003139

Resources

Type	Allocated	Used
Time	32.0h	82.8m
Nodes	1	
CPU cores	4	0.96
CPU cores by node	nl10803: 4	
Memory	100.0 GB	100.0 GB
Energy		◌
Electric car range equivalent		◌
CO2 emissions		◌

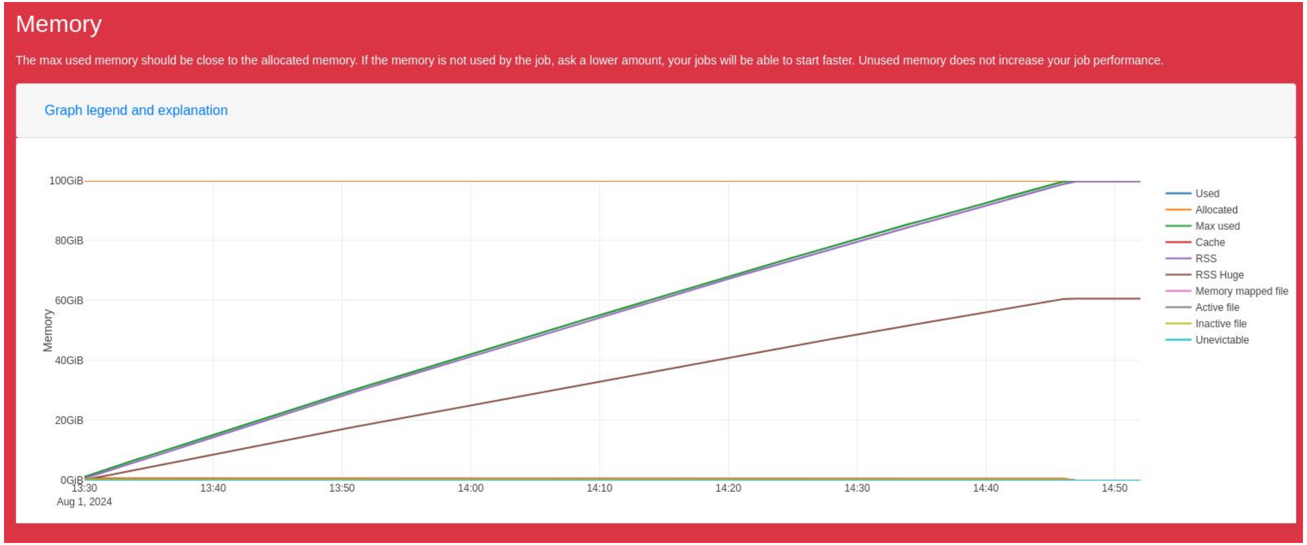
CPU waste within my own job



CPU

Ratio of cycles consumed on each CPU core by all processes in this job. This graph should be all filled up most of the time, if not, you can lower the cores requested to the scheduler. Unused cycles does not improve your job performance and will lower your priority when cores are wasted.

Less wasting of memory, but strange pattern



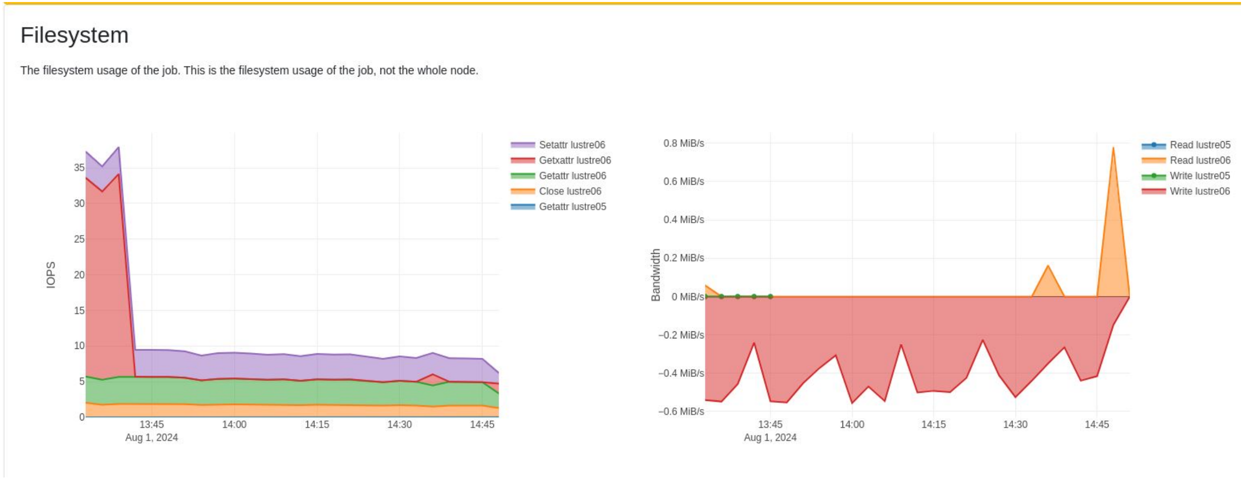
Memory

The max used memory should be close to the allocated memory. If the memory is not used by the job, ask a lower amount, your jobs will be able to start faster. Unused memory does not increase your job performance.

Graph legend and explanation

SHARCNET: Tyler Collins

Filesystem figures that help illustrate a bug



Filesystem

The filesystem usage of the job. This is the filesystem usage of the job, not the whole node.

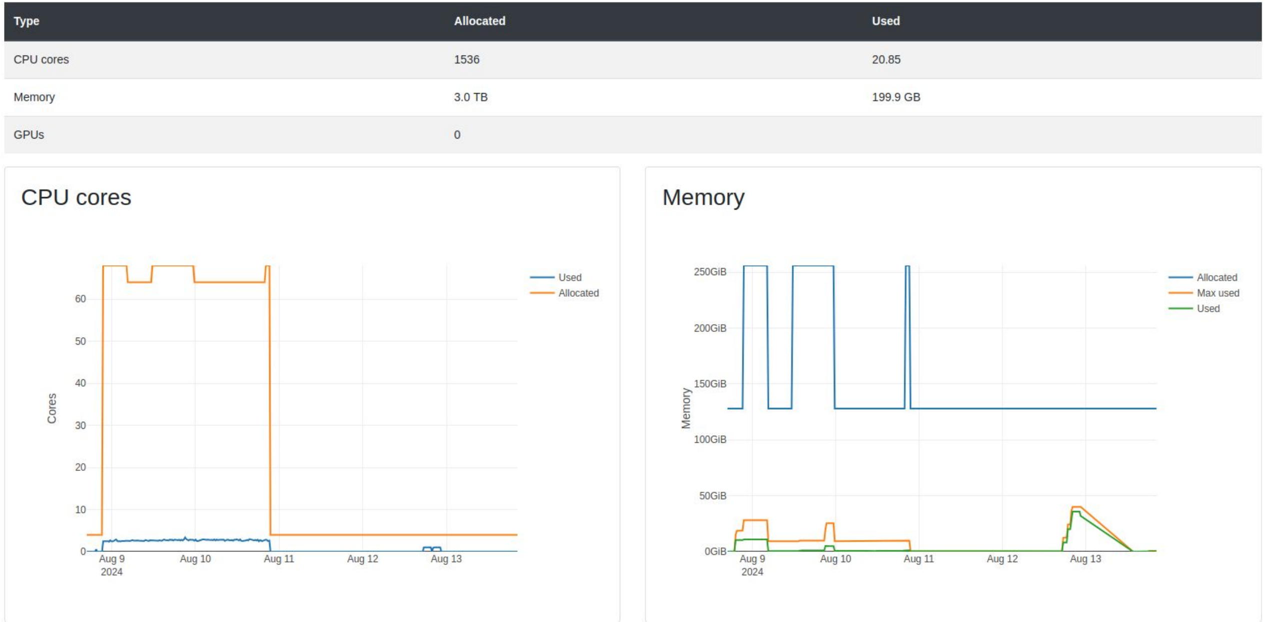
Some case studies

Let’s explore what your own jobs may show when looked at through the user portal

- These are handpicked real legacy cases
- Users were experiencing wait time they were concerned with
- Through eliminating waste, throughput increased significantly

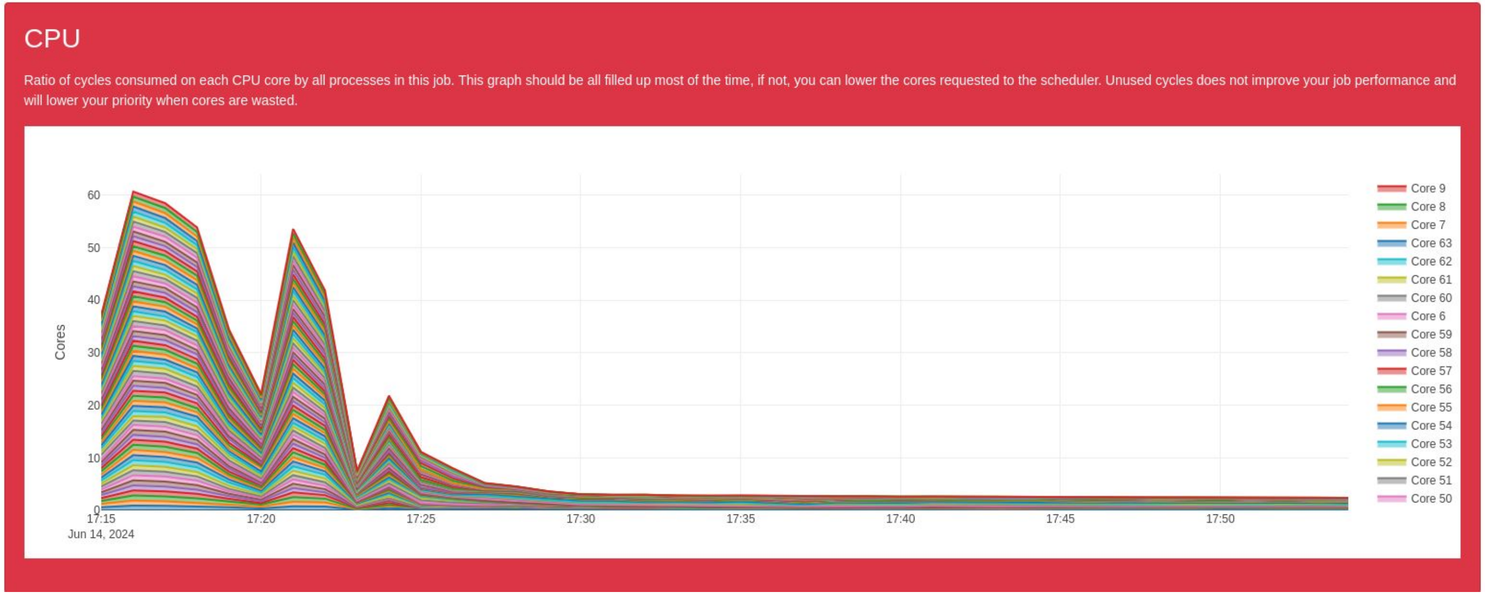
From these examples it should be there what would need to be adjusted to minimize waste

Core/memory waste: ~16-24 hour waits per job



Type	Allocated	Used
CPU cores	1536	20.85
Memory	3.0 TB	199.9 GB
GPUs	0	

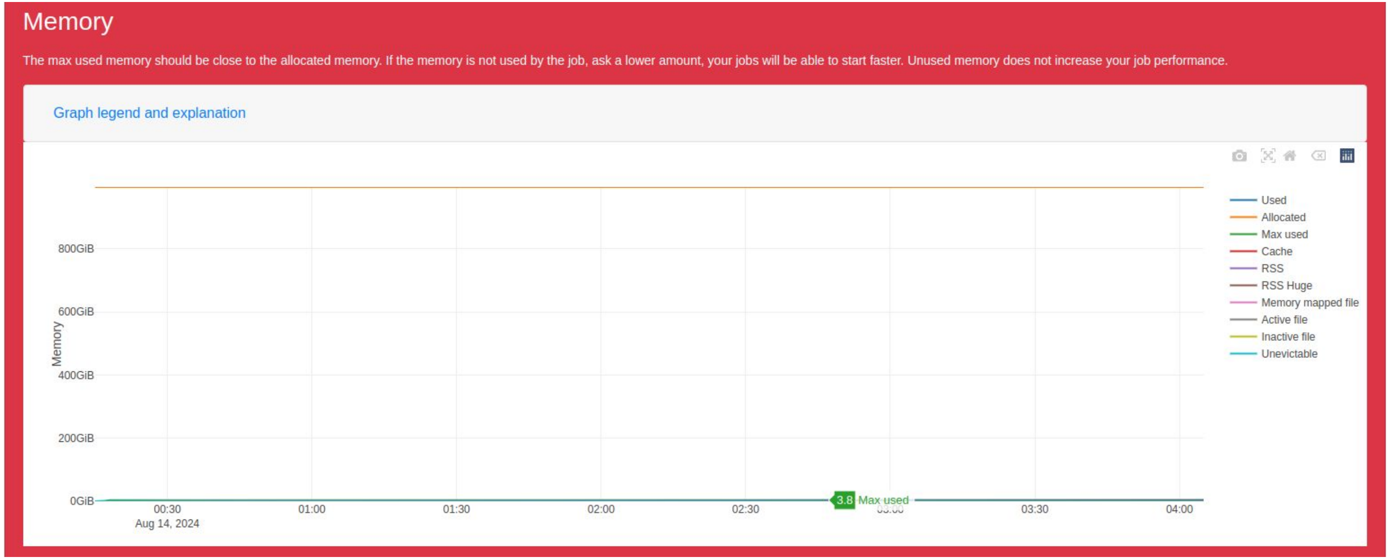
Core waste due to fall off: 3-day wait



CPU

Ratio of cycles consumed on each CPU core by all processes in this job. This graph should be all filled up most of the time, if not, you can lower the cores requested to the scheduler. Unused cycles does not improve your job performance and will lower your priority when cores are wasted.

Memory problems: over a week of waiting

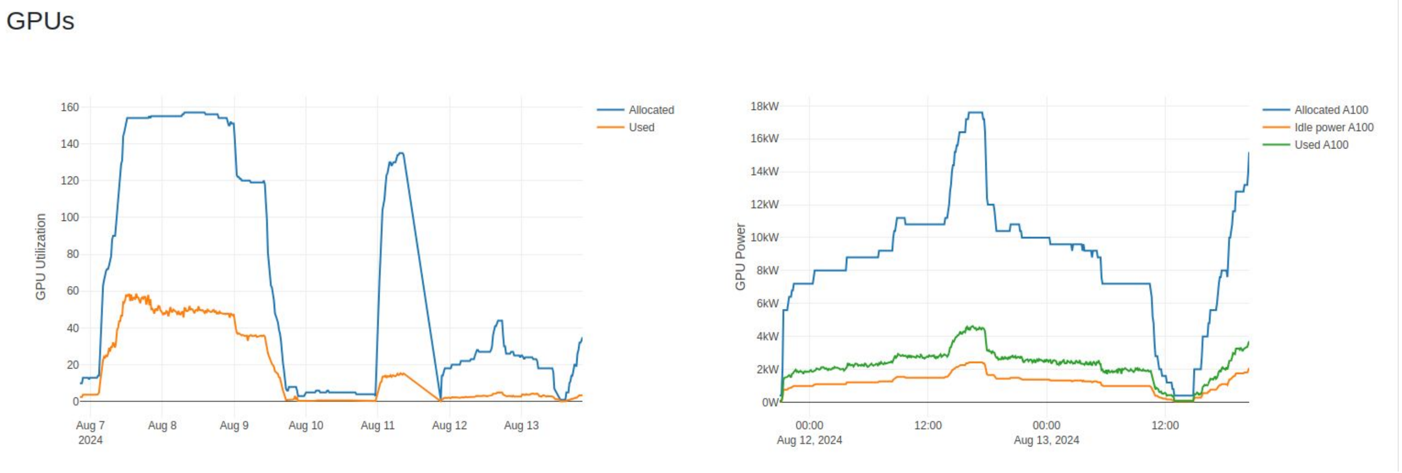


Memory

The max used memory should be close to the allocated memory. If the memory is not used by the job, ask a lower amount, your jobs will be able to start faster. Unused memory does not increase your job performance.

Graph legend and explanation

GPU waste: 100+ hours of waiting per week



SHARCNET: Tyler Collins

Userportal conclusions

This platform is not just good for finding where jobs are wasting resources, but can also help with debugging too

- Interactive jobs are supported
- Filtering by state
- For more advanced jobs, there are other resources like infiniband performance, etc

If anyone is feeling brave we can take a look at the accounts of some volunteers

Takeaways

- Eliminating waste via job shape optimization is the best way to reduce wait times
- The account usage portal offers a complete historical picture of usage
- The userportal offers an excellent in depth look at exactly what jobs are doing
- Users have an easier time than ever before to see exactly what their jobs are doing both within an account and within a job

Questions?