



Self-service Monitoring of HPC and Openstack Jobs for Users

Simon Guilbault (simon.guilbault@calculquebec.ca)

Quick overview of the portal

- Live demonstration of a staff view
- Public view of some information
 - <https://portail.narval.calculquebec.ca/>

The screenshot shows the Narval User portal. The top header bar is dark grey with 'Narval' on the left and 'Simon Guilbault (sigui4)' on the right. A left sidebar contains a menu with items: Home, Search, User summary, Job stats, Account stats, Cloud stats, Top, Notes, Nodes, External links, Documentation, Globus, JupyterHub, Other portals, Beluga, Graham, and Niagara. The main content area has a title 'User portal' and a section 'Cluster and services description'. The description text states that Narval is a general purpose cluster located at the École de technologie supérieure in Montreal, managed by Calcul Québec, and named in honour of the narwhal. It also mentions the cluster's scale (over 80,720 cores and 636 GPUs) and its connection speed (100Gb/s). Below the text is a list of links: Filesystems performance, Logins nodes, Scheduler, Scientific software, and Data transfer nodes.

Narval

Simon Guilbault (sigui4)

User portal

Cluster and services description

Narval is a general purpose cluster designed for a variety of workloads; it is located at the [École de technologie supérieure in Montreal](#) and is managed by [Calcul Québec](#). The cluster is named in honour of the [narwhal](#), a species of whale which has occasionally been observed in the Gulf of St. Lawrence.

This cluster is composed of over 80720 cores and 636 GPUs, all nodes are connected at 100Gb/s.

This portal is intended for our users, the menu on the left provides multiple tools to measure job performance and current cluster utilization

[Documentation](#)

- [Filesystems performance](#)
- [Logins nodes](#)
- [Scheduler](#)
- [Scientific software](#)
- [Data transfer nodes](#)

Quick overview of the portal



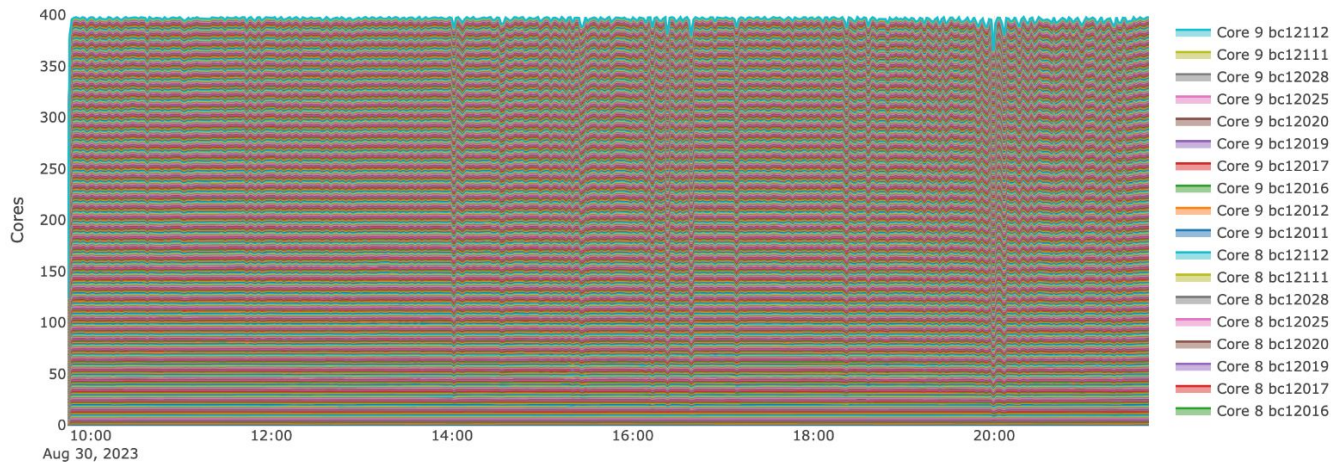
Beluga

Simon Guilbault (sigui4)

- Home
- User summary
- Job stats
- Account stats
- Top
- Notes
- External links ↕
- Documentation
- Globus
- JupyterHub
- Other portals ↕
- Narval
- Graham
- Niagara

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.



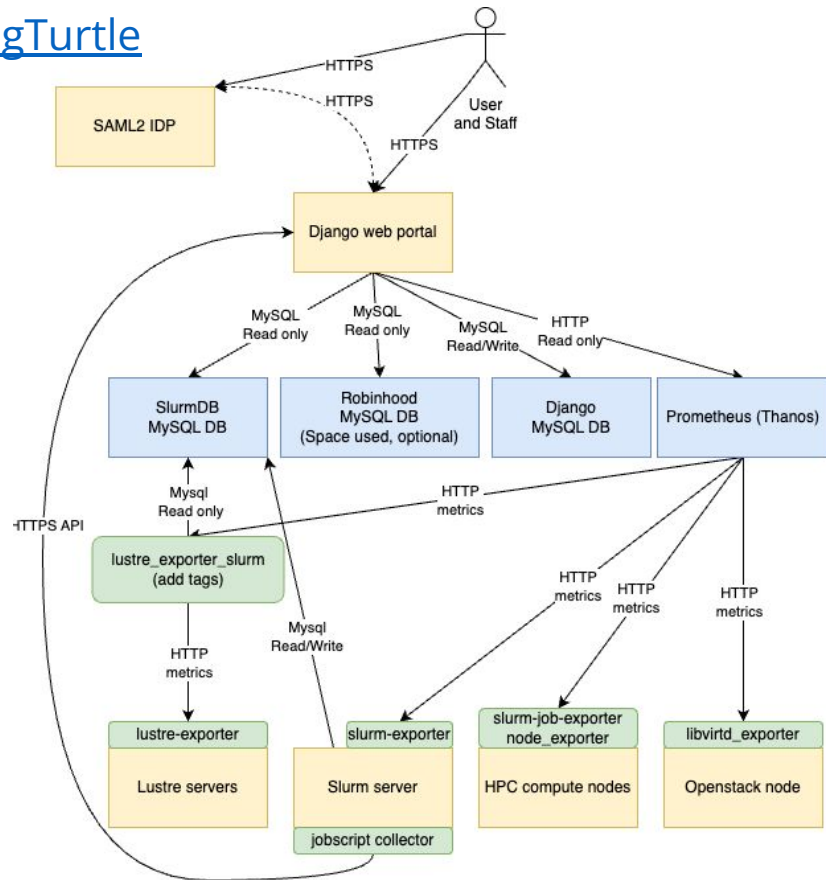
Memory

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](#)

Data flow and collectors

- The portal: github.com/guilbaults/TrailblazingTurtle
- [slurm-job-exporter](#)
- [node exporter](#)
- [slurm-exporter](#)
- [lustre exporter](#) + [lustre exporter slurm](#)
- [redfish exporter](#) (power by node)
- [libvirt exporter](#) (openstack)
- [pcm-sensor-server](#) (Intel IPC/NUMA/Mem)



Prometheus

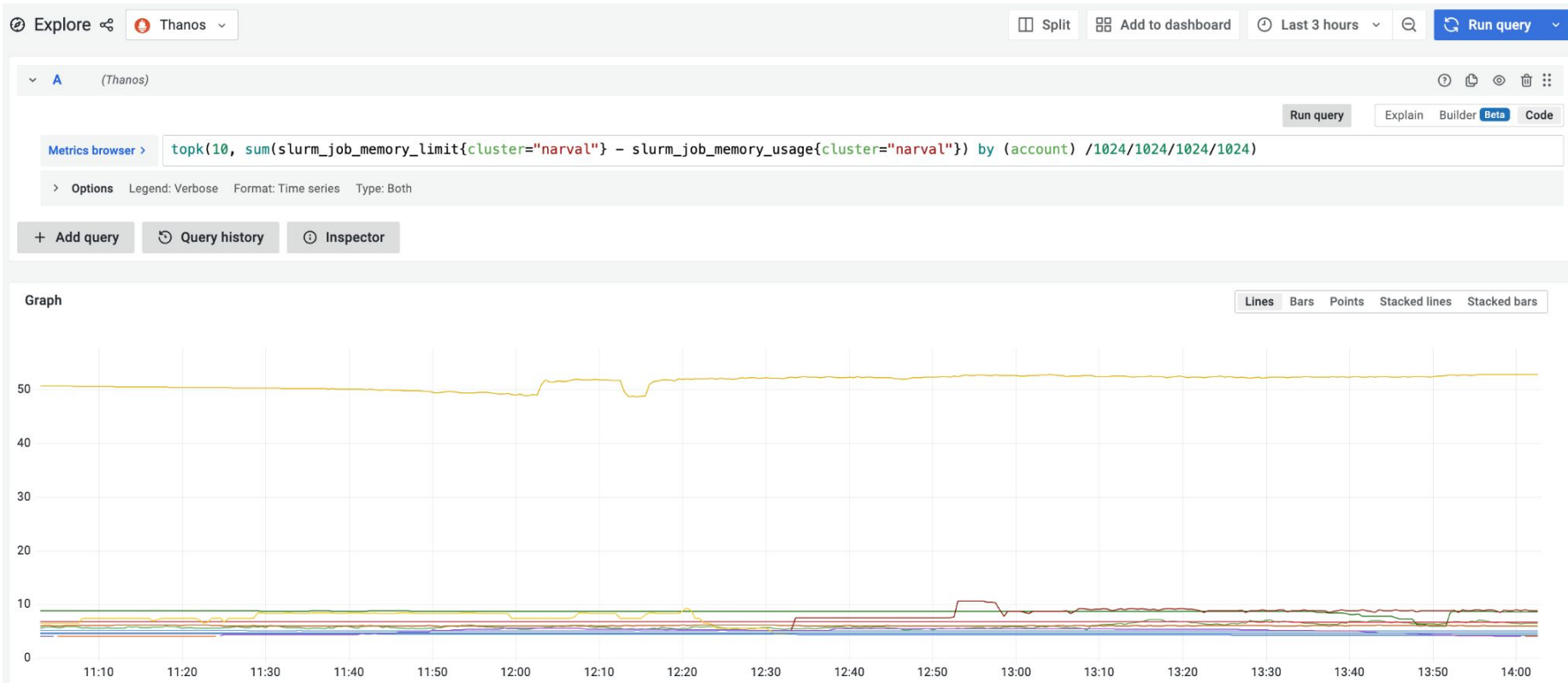
- 1866 nodes monitored
- 200k metrics per second
- 1 VM per cluster
 - 50GB of ram
 - 6 cores
 - 300GB of local disk
 - 10 IOPS
- Cardinality is not a issue for jobstats
 - Recorder rules to generate sums

Thanos

- Plugin over Prometheus for archival and multiple endpoints
- 3.5+ clusters monitored
- 27TB in S3 (ceph)
- 6 months of retention for everything
- Rewrite after 6 months to remove some stats
- Downsampling is disabled
 - Downsampling increase storage req.
 - Too many small jobs
 - Queries over the last year are rare

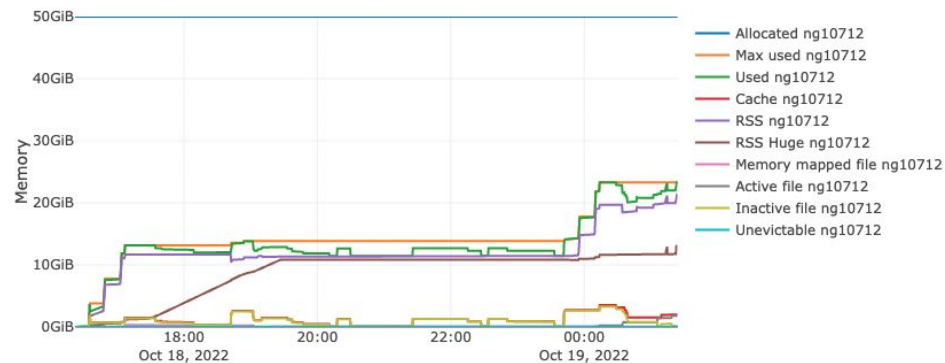
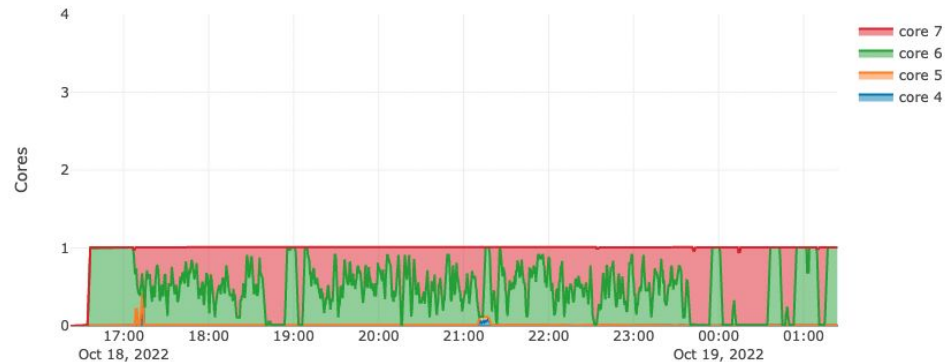
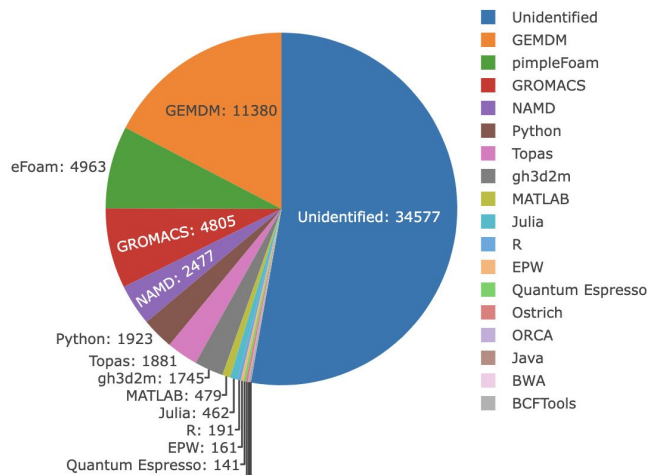
Initial prototype and exploration in Grafana

- Simple query to get the 10 worst memory waster by user
 - Not user friendly and only available to sysadmins



slurm-job-exporter

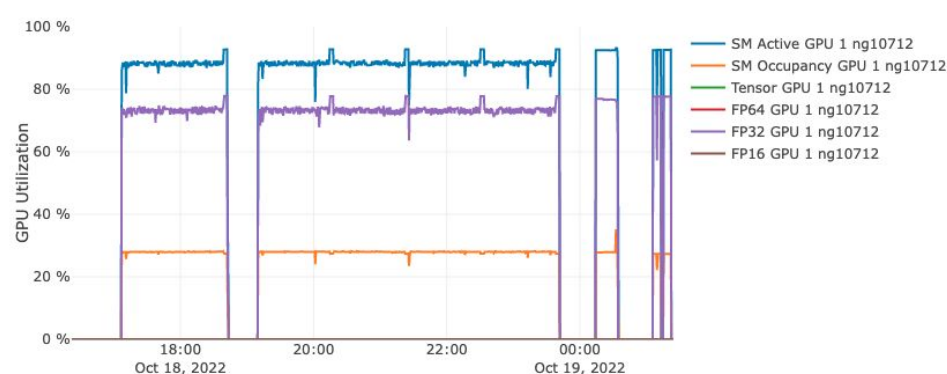
- cgroups statistics
 - CPU
 - Memory
 - Application name
 - Processes and threads count



slurm-job-exporter

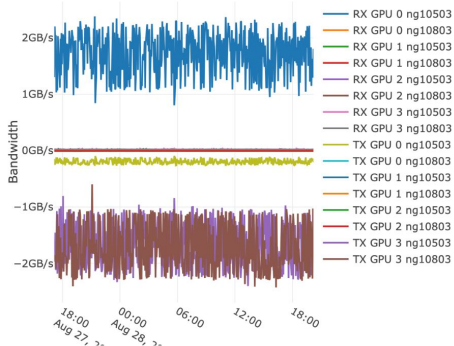
Use Nvidia DCGM, include MIG stats

- SM active/occupied
- FP64/32/16/Tensors
- Memory bandwidth
- PCIe and Nvlink bandwidth



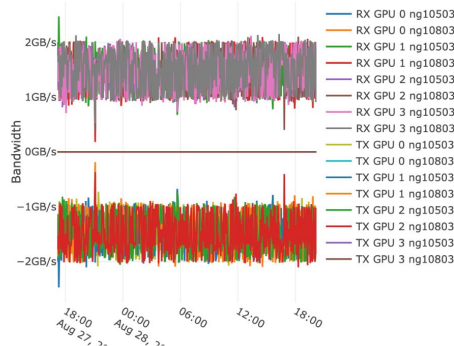
GPU PCIe bandwidth

Note that this is from the perspective of the GPU



GPU Nvlink bandwidth

Note that this is from the perspective of the GPU

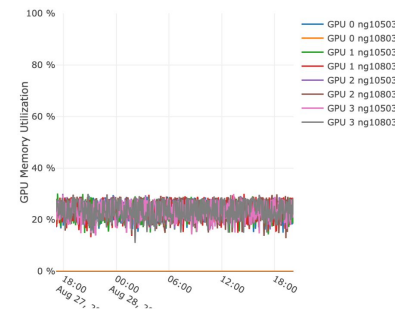


GPU Memory used



GPU Memory access cycle used

The ratio of cycles the device memory interface is active sending or receiving data



node_exporter

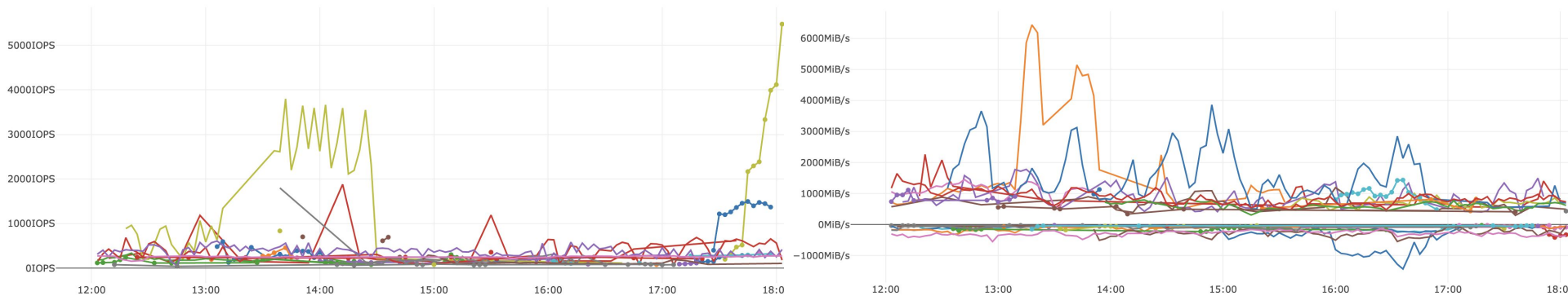
- Local disk
 - IOPS
 - Bandwidth
 - Used space
 - Network
 - Ethernet and Infiniband
 - Unix load
 - CPU
 - Memory + NUMA
 - ZFS
-
- [Grafana node-exporter full](#)

redfish_exporter

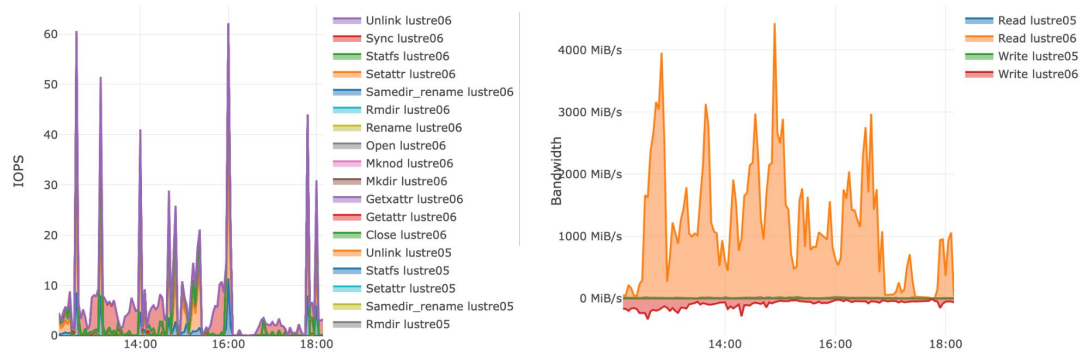
- Power
 - iDRAC does some averaging over time
- Fans
- Temperature

Lustre

Top users for bandwidth and IOPS



Usage for a user or a job



Automated Job analysis

- Regex on job script
 - Stats from prometheus
 - Application name
- ```
#SBATCH --ntasks=96 # number of MPI processes
module load StdEnv/2020 gcc/9.3.0 openmpi/4.0.3
gromacs/2020.4
gmx grompp -f $mdp/emin_$LAMBDA.mdp
gmx mdrun -v -deffnm emin$LAMBDA -nt 2
sleep 10
```

Less than half the CPU compute cycle were used

This job is using multiple nodes

This job is running on average 7.5 threads on 16 cores, the cores might be underused

Application  
/cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/Core/py  
used 6.5 cores on average

Show submitted job script

This job is using multiple nodes

Line 27: GROMACS preprocessor should be used on a login node

Line 29: GROMACS is used without srun or mpirun/mpiexec

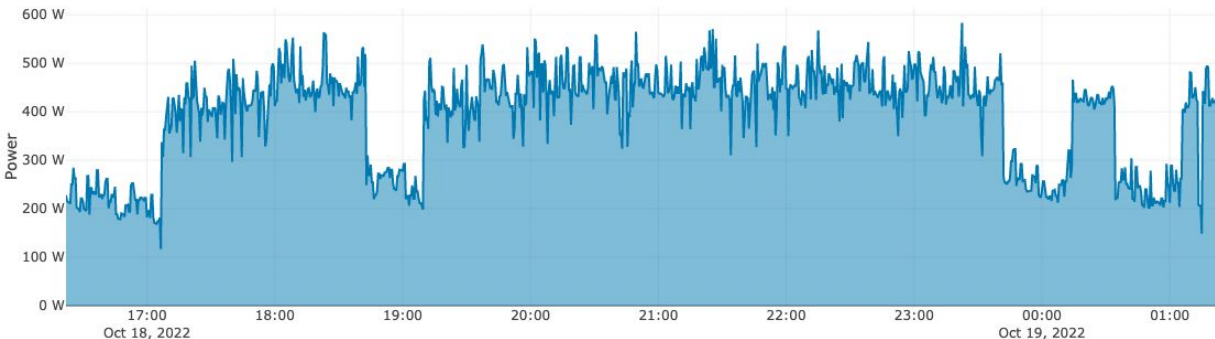
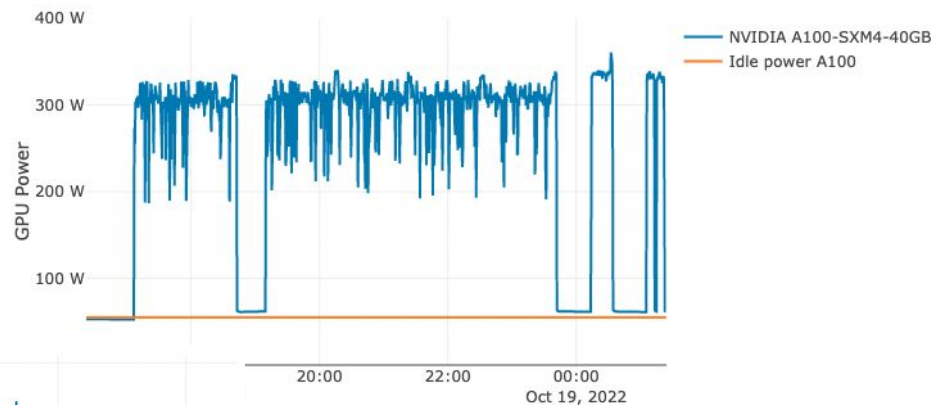
Line 29: GROMACS is used with -nt 2 instead of -nt 96

Line 29: Multiple nodes are used without the MPI binary

Line 31: sleep command is used

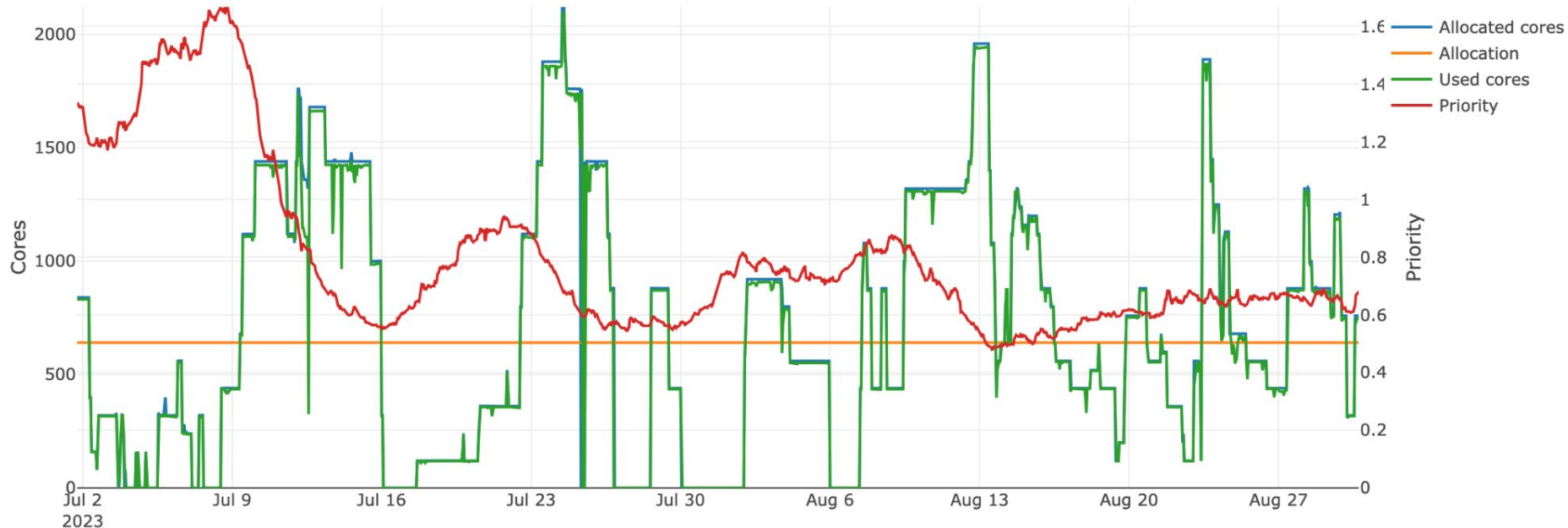
# Power measurement

- Power by node and by GPU
- Allocate power of a shared nodes to the jobs running on it
  - based on % of cpu cores
  - by the GPU used



# Priority and group use

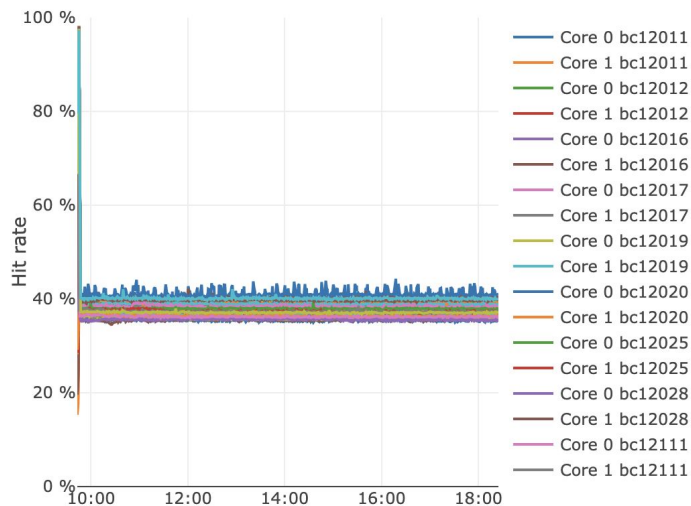
Priority (levelFS) is around 0.6 because this group is using more than their allocation



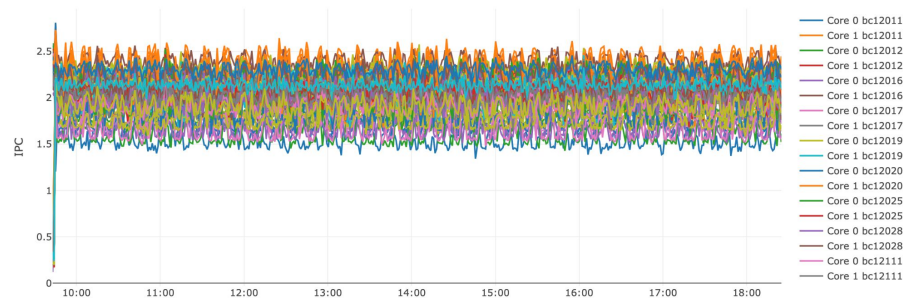
# Intel pcm-sensor-server

AMD CPU are missing some counters for memory bandwidth and other metrics

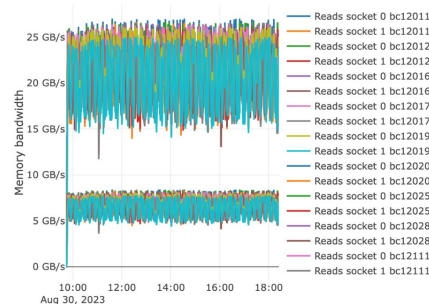
## L3 cache hit rate



## Instructions per cycle



## Memory bandwidth

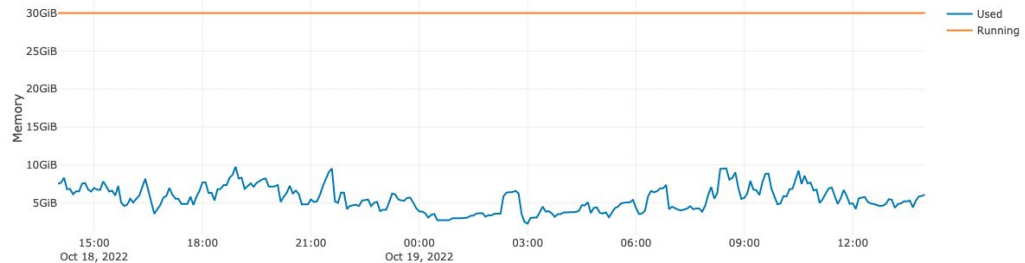
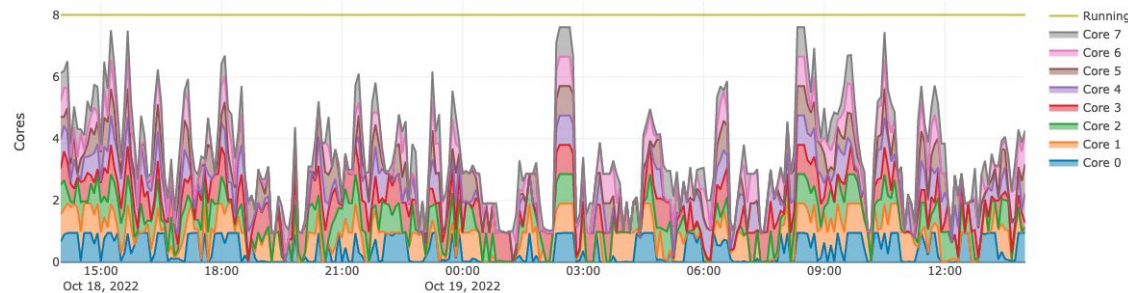


## CPU interconnect bandwidth



# Openstack

- Overall utilization
  - by project
  - by VM
- Cores
- Memory
  - Allocated
  - Balloon to see used
- Network
  - Bandwidth
  - Packets
- Block devices
  - Bandwidth
  - IOPS



# Future improvements

- Better automated job analysis
  - Simple score for each job/user/slurm account
  - Feedback on job script
- Automatic alerts and killing jobs
  - What count as not using a GPU correctly ?
- Improving the annual allocation process
- Improving the next RFP based on the resources being used
- Node fail additional informations from syslog
  - Tell the user the node died because of a DIMM
- Analyse compiled applications by used CPU time
  - Language
  - Compiler
- Clarification in MIG stats



# Questions ?

- <https://github.com/guilbaults/TrailblazingTurtle/>