Slurm status and news from the Nordics

2015-03-27, Hepix spring 2015, Oxford

norden

NordForsk

Nordic e-Infrastructure Collaboration

Overview

- SLURM usage
- Memory counting
- Backfill
- Future

Slurm usage

- The production use batchsystem for all sites but one in NDGF
  - Last site in preproduction, switching “any month now”
- Also the batch system of choice by the HPC sites in the nordics
- Fair bit of overlap between these, since most of the NDGF CEs are connected to HPC clusters
  - WLCG usage being a middle-sized user with “weird jobs”
  - But not unique in doing smaller high throughput computing

Memory – Virtual Set Size (VSS)

- VSS (reported as VSZ from ps) is the total accessible address space of a process. This size also includes memory that may not be resident in RAM like mallocs that have been allocated but not written to or mmap()ed files on disk, etc.
- VSS is of very little use for determing real memory usage of a process.
- VSS is sometimes called Address Space (AS)

Resident Set Size (RSS)

- The non-swapped physical memory a task is using
  - RSS is the total memory actually held in RAM for a process. RSS can be misleading, because it reports the total all of the shared libraries that the process uses, even though a shared library is only loaded into memory once regardless of how many processes use it.
  - It also counts all other shared pages, such as the copy-on-write pages still shared with the parent after fork(), an important usecase for current LHC multicore usage.
- RSS is not an accurate representation of the memory usage for a single process

Proportional Set Size (PSS)

- PSS differs from RSS in that it reports the proportional size of its shared pages
  - Ex: if three processes all use a shared library that has 30 pages, that library will only contribute 10 pages to the PSS that is reported for each of the three processes.
- PSS is a very useful number because when the PSS for all processes in the system are summed together, that is a good representation for the total memory usage in the system.
- When a process is killed, the shared libraries that contributed to its PSS will be proportionally distributed to the PSS totals for the remaining processes still using that library. In this way PSS can be slightly misleading, because when a process is killed, PSS does not accurately represent the memory returned to the overall system.

Unique Set Size (USS)

- USS is the total private memory for a proces, i.e. that memory that is completely unique to that process.
- USS is an extremely useful number because cost of running a particular process. When a process is killed, the USS is the total memory that is actually returned to the system.
- USS is the best number to watch when initially suspicious of memory leaks in a process.

What does Slurm do?

- The slurm monitoring process counts Sum(RSS) for all processes in a job
- And kills the job if this sum is larger than the requested memory
- Even if you are using cgroups for memory limit enforcement!

What does Atlas do in multicore jobs?

- Load lots of data and setup large data structures
- fork()
- Use the data structures above read-only
- This makes use of the copy on write semantics for Linux memory to only have one physical copy of those pages, saving lots of RAM
- But Slurm still counted whis with Sum(RSS) and killed jobs that were well within the actual use

What have we done?

- Changed Sum(RSS) to Sum(PSS)
- Introduced a flag to disable Slurm's killing and rely only on cgroups to enforce memory limits
- Available as an option in upcoming versions of Slurm (15.08..)
- Patch available on request for 14.03 versions.
- No longer have to artificially increase the memory requirements in order to prevent killing, enabling tighter packing of jobs on the nodes!

Backfill in Slurm for HPC+Grid

Backfilling grid jobs on an HPC cluster

- An HPC cluster with a mix of jobs that include long and wide MPI jobs have lots of “holes” in the schedule
- Small grid jobs (1 core to 1 node) would fit nicely in these, making the “percent cores running jobs” go up
- Short (walltime requirement) jobs would have the best chance of getting scheduled!

Problems with the Slurm backfil scheduler

- Slurm is a pretty strict priority scheduler
  - Schedules all jobs in a priority order
  - No separate “backfill” stage that just starts jobs that can start, but places all the jobs with reservations at their optimal starting point
- The scheduler gets cancelled after a few hundred jobs
  - Never gets deep enough in the queue to find the small jobs that can actually fit somewhere if you are unlucky

Why does Slurm cancel backfill scheduling

- On any state change
  - Cluster change
  - Node change
  - Reservations
  - Job list changes
    - New job submitted, job finished, job started, job cancelled, etc
  - Job steps
    - Start, stop, etc

When does it need to be canceled?

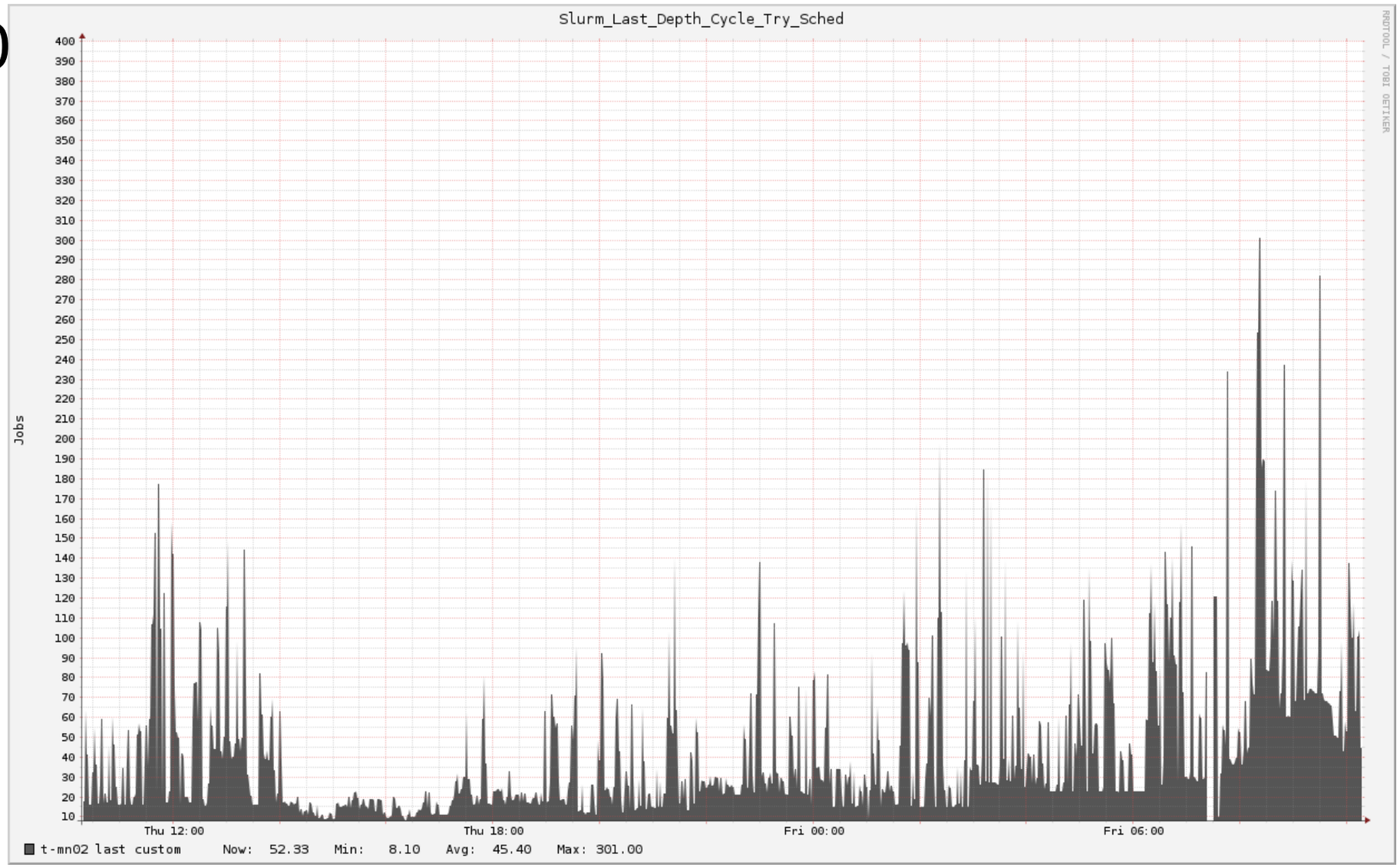
- Cluster, node, reservations changes?
- Job starts?
- Job step starts?
- Job step ends?
- Job ends?
- Job submitted?

Problems and Solutions

- The issue comes from having a large system
  - Lots of jobs
  - Lots of changes
- What have we done?
  - Only job endings cancel the backfill out of the job state changes

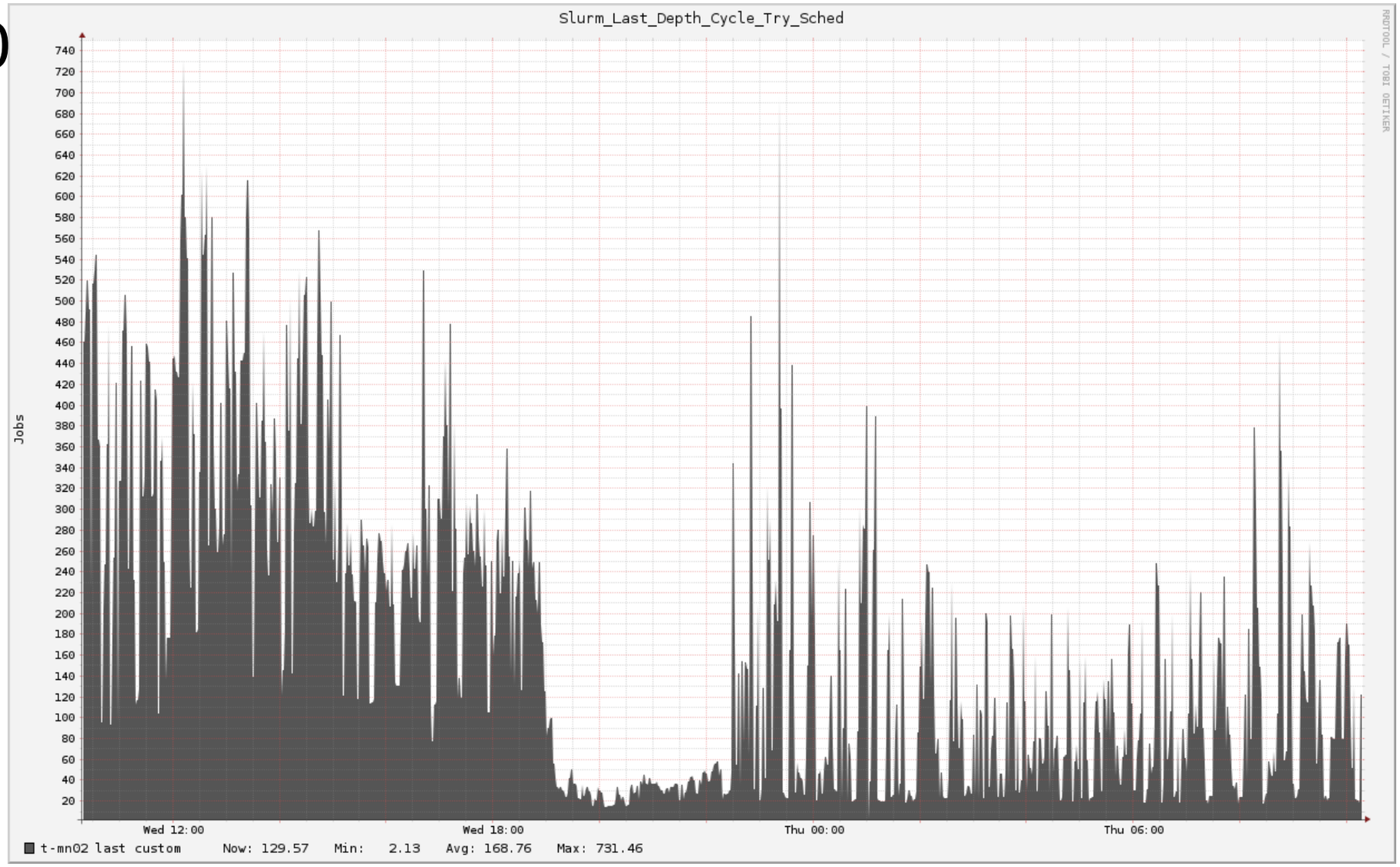
Before – how deep in the queue we got

400



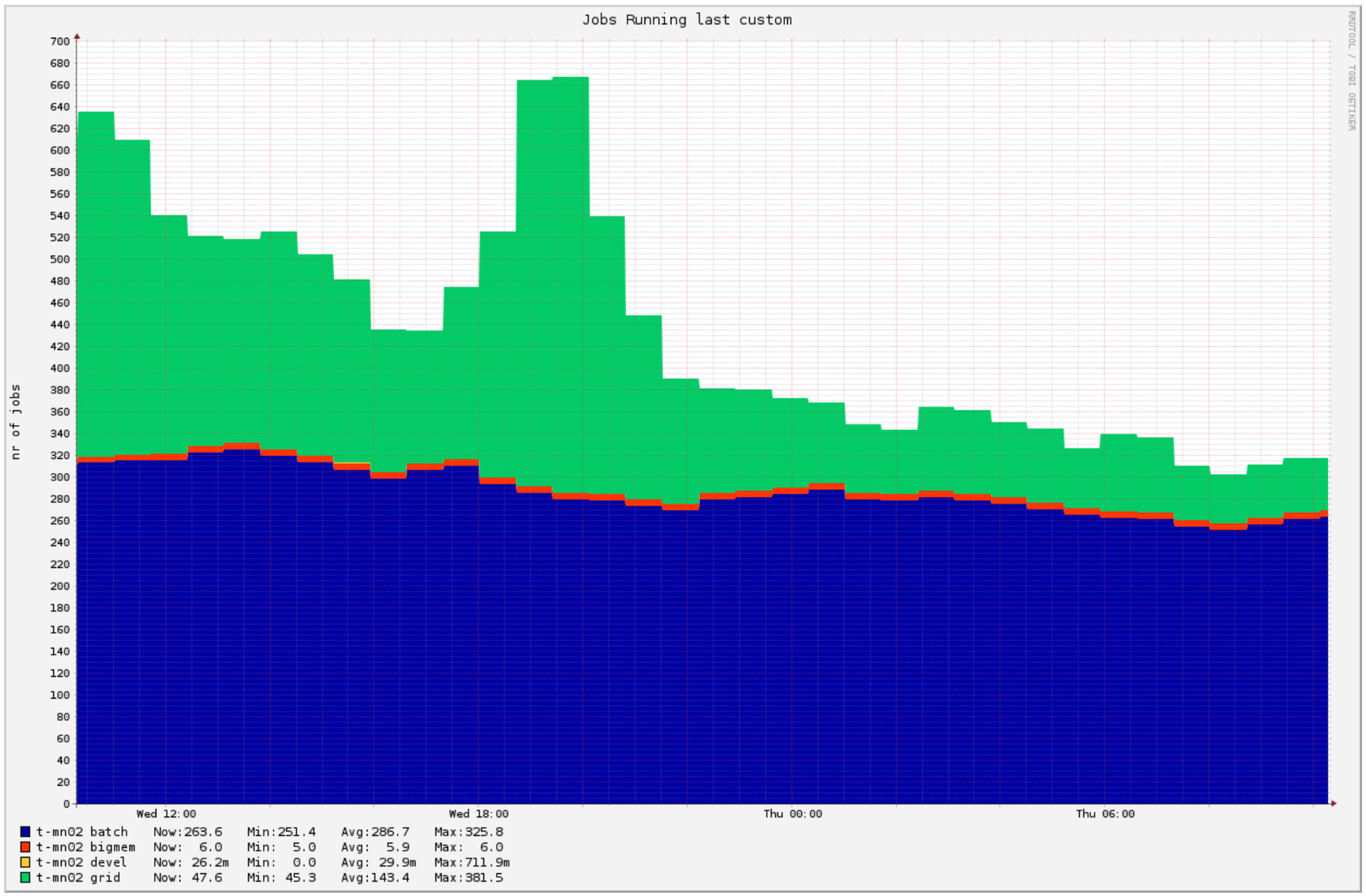
After – how deep in the queue we got

740



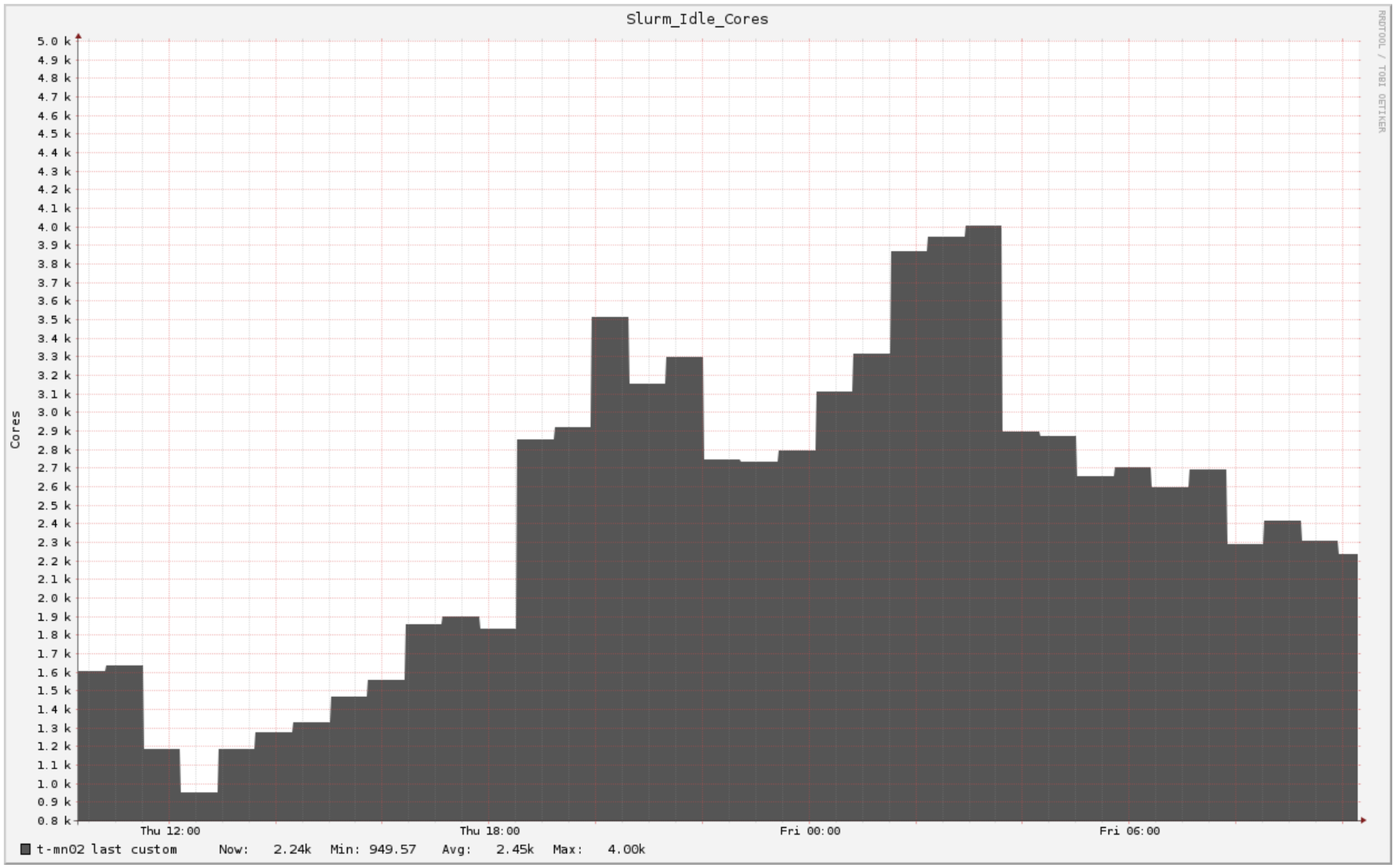
After – job type overview

700



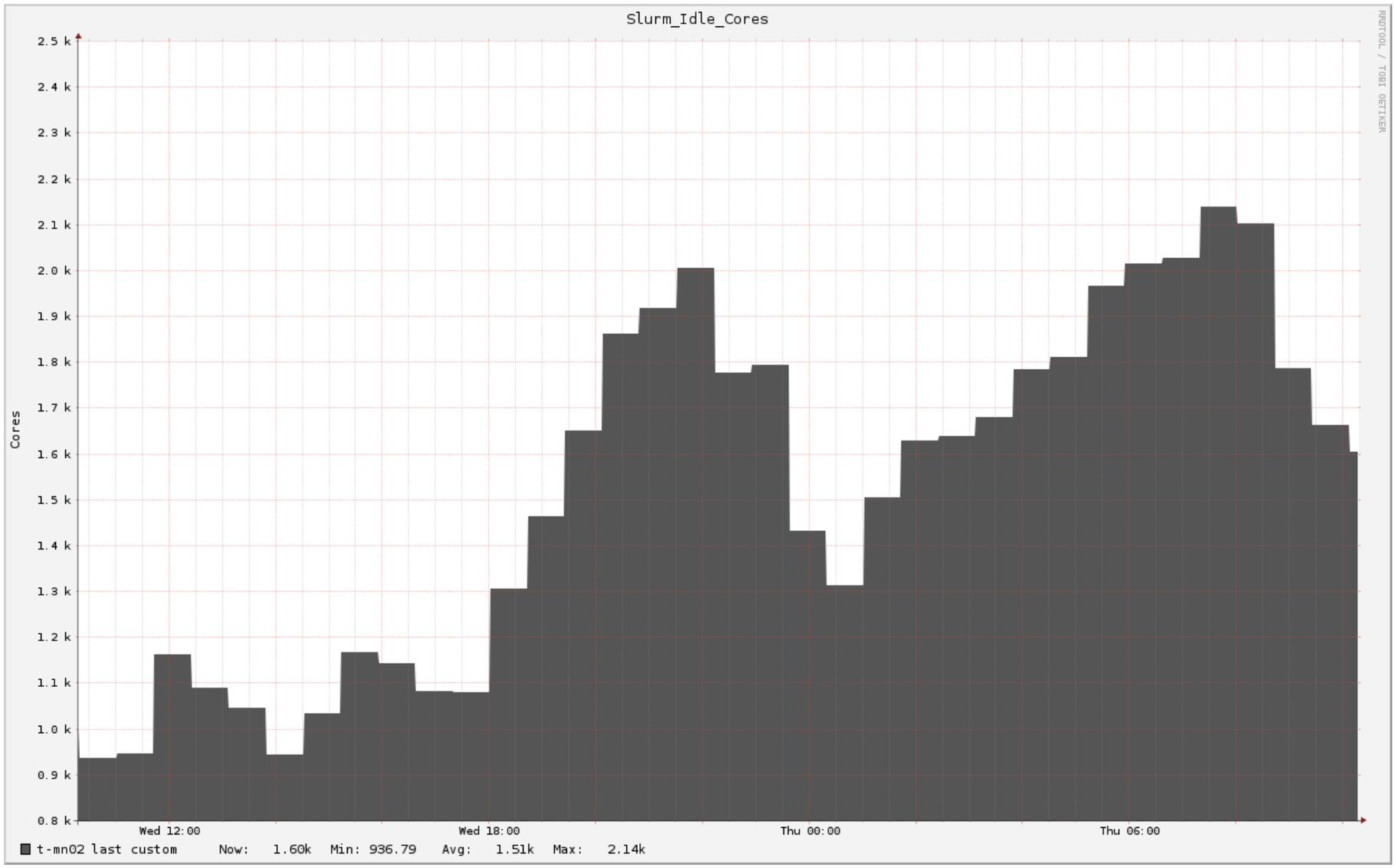
Before – Idle cores

5.0k



After – Idle cores

2.5k



Ideas for future work

- Different per-user backfill depth for different partitions
- Two scheduling steps
  - One priority scheduling step that can stop as soon as every slot has a reservation at some point in the future
  - One backfill scheduling step that just looks at the holes in the schedules and starts the jobs that can start now
- More accurate walltime estimates for WLCG jobs
  - Lots of jobs finish long before their reserved time
- Preemtable jobs
  - Most profitable if you can write out each jobstep (simulated event?) and recover from wherever the job was foribly ended (or handle restarts with REQUEUE)
  - Possibly gains of 2-5k cores within ND cloud “for free” if this could be made to work reliably

And now for a message from our sponsors

- NDGF-T1 is a distributed site over 6 sites in the Nordics
  - A central site might be more cost-efficient
  - But there are tradeoffs in resiliency etc
  - Sharing competence and resource with HPC at the moment
- Looking for someone independent yet knowledgeable
- 3 month paid contract available for investigation and report writing
- Full job posting and details on http://www.neic.no
- Or talk to me

NordForsk

Nordic e-Infrastructure Collaboration

Questions?