



SLURM @BIANCA

Diana Iușan

*Application expert and
training coordinator UPPMAX*



UPPSALA
UNIVERSITET

More Slurm and other advanced UPPMAX techniques

- A closer look at Slurm
- Using the GPUs
- Job efficiency with the **jobstats** tool
- Advanced job submission



The **Slurm** Workload Manager

- Free, popular, lightweight
- Open source:
<https://slurm.schedmd.com>
- available at all NAISS centra
- UPPMAX Slurm userguide:

[https://www.uppmax.uu.se/support/
user-guides/slurm-user-guide/](https://www.uppmax.uu.se/support/user-guides/slurm-user-guide/)



More on **sbatch**

- Recap:

sbatch	-A sens2023598	-t 10:00	-p core	-n 10	my_job.sh
Slurm batch	Project name	Maximum runtime	Partition ("job type")	#cores	job script



More on time limits

- **-t dd-hh:mm:ss**
- 0-00:10:00 = 00:10:00 = 10:00 = 10
- 0-12:00:00 = 12:00:00
- 3-00:00:00 = 3-0
- 3-12:10:15



Job wall time

- Q: When you have no idea how long a program will take to run, what should you book?
 - A: very long time, e.g. 10-00:00:00
- Q: When you have an idea of how long a program would take to run, what should you book?
 - A: overbook by 50%



More on partitions

- **-p core**

- “core” is the default partition
- ≤ 16 cores on Bianca
- a script or program written without any thought on parallelism will use 1 core

- **-p node**

- if you wish to book full node(s)



Quick testing

- The “devel” partition
 - max 2 nodes per job
 - up to 1 hour in length
 - only 1 at a time
 - **-p devcore, -p devel**
 - Any free nodes in the devel partition? Check status with
 - `sinfo -p devel`
 - **`jobinfo -p devel`**



Quick testing

- Any free nodes in the devel partition? Check status with
 - `sinfo -p devel`
 - **`jobinfo -p devel`**
 - more on these tools later
- High priority queue for short jobs
 - 4 nodes
 - up to 15 minutes
 - **`--qos=short`**



Debugging or complicated workflows

- Interactive jobs
 - handy for debugging a code or a script by executing it line by line or for using programs with a graphical user interface
 - `salloc -n 80 -t 03:00:00 -A sens2023598`
 - **`interactive`** `-n 80 -t 03:00:00 -A sens2023598`
 - up to 12 hours
 - useful together with the `--begin=<time>` flag
 - `salloc -A sens2023598 --begin=2022-02-17T08:00:00`

asks for an interactive job that will start earliest tomorrow
at 08:00



Parameters in the job script or the command line?

- Command line parameters override script parameters
- A typical script may be:

```
#!/bin/bash
```

```
#SBATCH -A sens2023598
```

```
#SBATCH -p core
```

```
#SBATCH -n 1
```

```
#SBATCH -t 24:00:00
```

- Just a quick test:

```
sbatch -p devcore -t 00:15:00 jobscript.sh
```



Hands-on #1:

`sbatch/jobinfo`

- login to Bianca
- find out which projects you're a member of using projinfo
- submit a short (10 min) test job; note the job ID
- find out if there are any free nodes in the devel partition
- submit a new job to use the devel partition
- write in the HackMD when you're done



Memory in core or devcore jobs

- $-n \times$
- Bianca: 8GB per core
- Slurm reports the available memory in the prompt at the start of an interactive job



More flags

- `-J <jobname>`
- email:
 - `--mail-type=BEGIN,END,FAIL,TIME_LIMIT_80`
 - `--mail-user` - Don't use. Set your email correctly in SUPR instead.
- out/err redirection:
 - `--output=slurm-%j.out` *and* `--error=slurm-%j.err`
by default, where `%j` will be replaced by the job ID
 - `--output=my.output.file`
 - `--error=my.error.file`



More flags

- Memory
 - `-C thin / -C 128GB`
 - `-C fat / -C 256GB / -C 512GB`
- Dependencies: `--dependency`
- Job array: `--array`
- More at <https://slurm.schedmd.com/sbatch.html>
 - or just `man sbatch`
 - not all options work on all systems!



Monitoring jobs

- **jobinfo** - a wrapper around **squeue**
 - lists running and pending jobs
 - `jobinfo -u username`
 - `jobinfo -A sens2023598`
 - `jobinfo -u username --state=running`
 - `jobinfo -u username --state=pending`
- One may also use the **squeue** command.
- **bianca_combined_jobinfo** (queued jobs of all projects)



Monitoring and modifying jobs

- **scontrol**

- `scontrol show job jobid`

- possible to modify the job details after the job has been submitted; some options, like maximum runtime, may be modified (=shortened) even after the job started

- `scontrol update JobID=jobid QOS=short`

- `scontrol update JobID=jobid TimeLimit=1-00:00:00`

- `scontrol update JobID=jobid NumNodes=10`

- `scontrol update JobID=jobid Features=mem1TB`



When a job goes wrong

- **scancel**

- *jobid*
- *-u username* - to cancel all your jobs
- *-t state* - cancel pending or running jobs
- *-n name* - cancel jobs with a given name
- *-i* - ask for confirmation



Priority

- Roughly:
- The first job of the day has elevated priority
- Other normal jobs run in the order of submission (subject to scheduling)
- Projects exceeding their allocation get successively into the lower priority category
- Bonus jobs run after the jobs in the higher priority categories



Priority

- In practice:
 - submit early = run early
 - bonus jobs always run eventually, but may need to wait until the night or weekend
- In detail:

<https://www.uppmax.uu.se/support/faq/running-jobs-faq/your-priority-in-the-waiting-job-queue/>



Hands-on #2:

`sbatch/squeue/scancel/scontrol/jobinfo`

- submit a new job; note the job ID
- check all your running jobs
- what is the priority of your recently-submitted job?
- submit a new job to run for 24h; note the job ID
- modify the name of the job to “wrongjob” and the maximum runtime to 7days, for example
- cancel your job with name “wrongjob”





Determining job efficiency

- **jostats** - custom-made UPPMAX tool



Job efficiency

- **jobstats** - a tool in the fight for productivity
 - it works only for jobs longer than 5-15 minutes
 - `-r jobid` - check running jobs
 - `-A project` - check all recent jobs of a given project
 - `-p jobid` - produce a CPU and memory usage plot
 - <http://docs.uppmax.uu.se/software/jobstats/>



Hands-on #3: jobstats

- Generate jobstats plots for your jobs
 - Firstly, find some job IDs from this month
 - `finishedjobinfo -m username`
 - Write down the IDs from some interesting jobs.
 - Generate the images:
`$ jobstats -p ID1 ID2 ID3`
- Look at the images
 - `$ eog *png &`



Hands-on #3: j o b s t a t s

- Which of the plots
 - Show good CPU or memory usage?
 - Indicate that the job requires a fat node?



11217530 COMPLETED on rackham end: 2019-12-17T12:35:44 runtime: 1-02:16:12

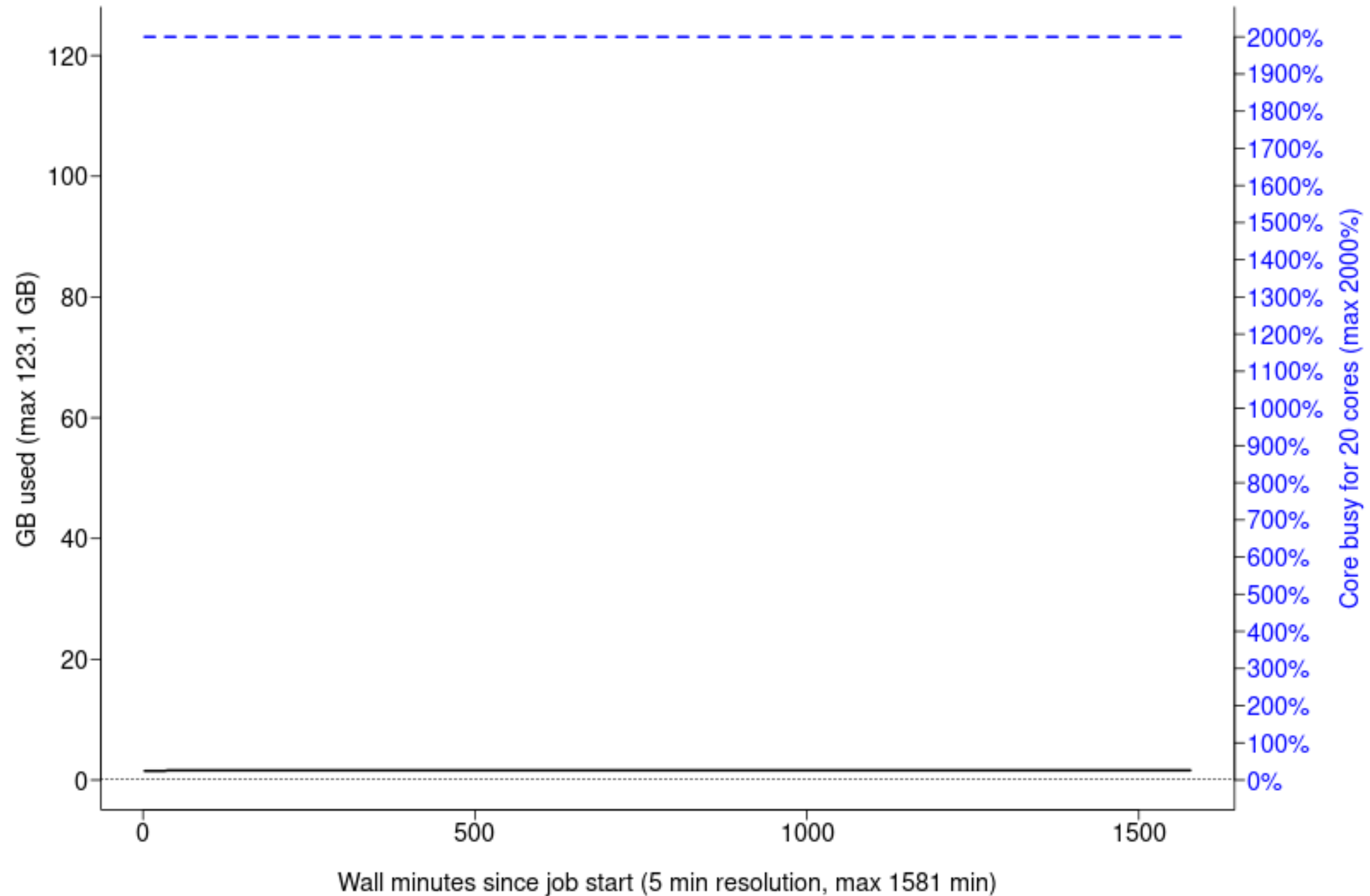
User:

Proj: snic2019-1-12

Jobname: fesn_x

Flags: none

r132



8804061 COMPLETED on milou end: 2016-10-10T20:30:26 runtime: 04:36:33

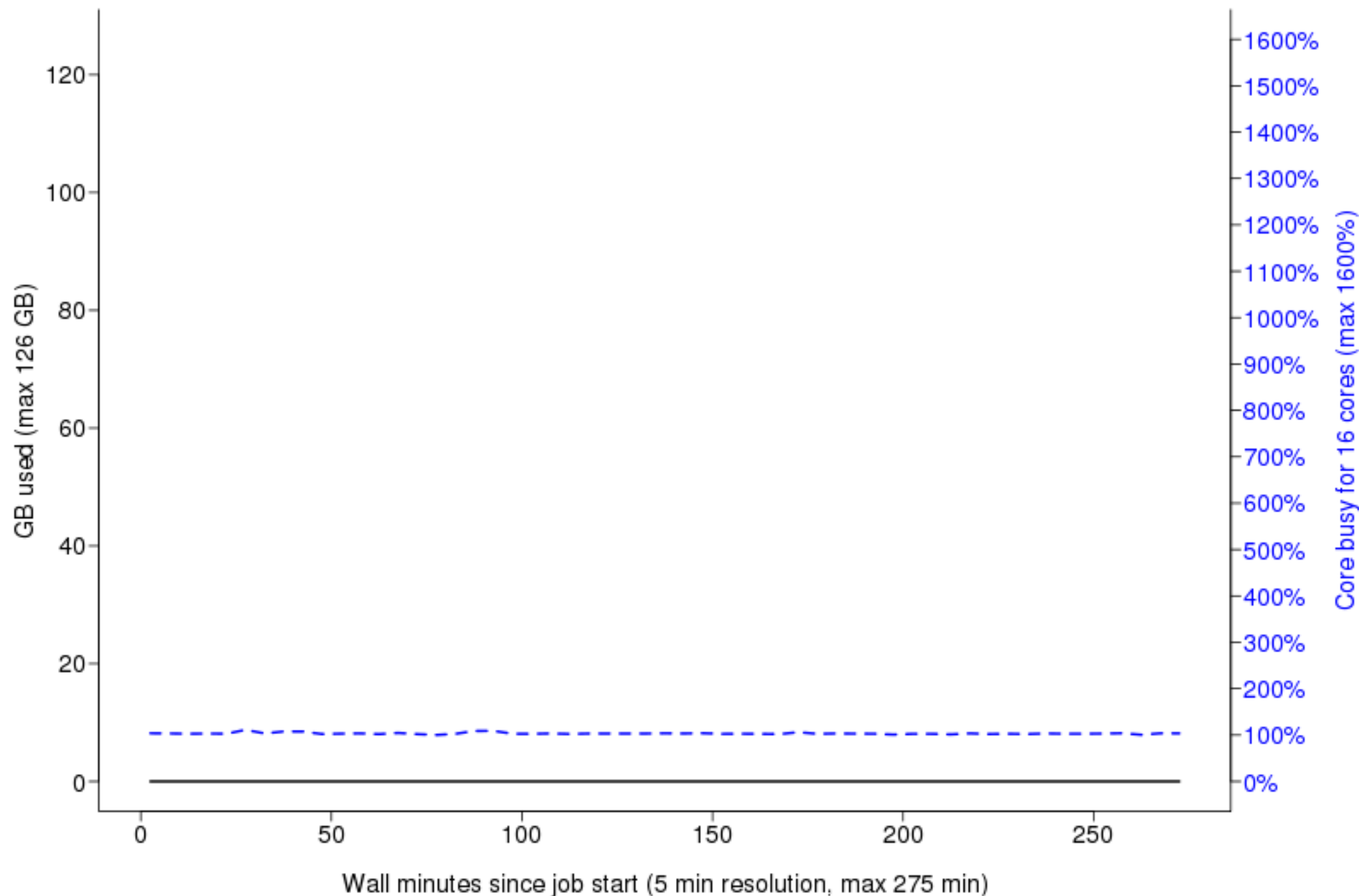
User:

Proj: b2015110

Jobname: nf-MergeBam_(2)

overbooked:12%, !!half_overbooked, !!severely_overbooked,
cores_overbooked:16:2, mem_overbooked:126:0, core_mem_overbooked:15.8:0

m102



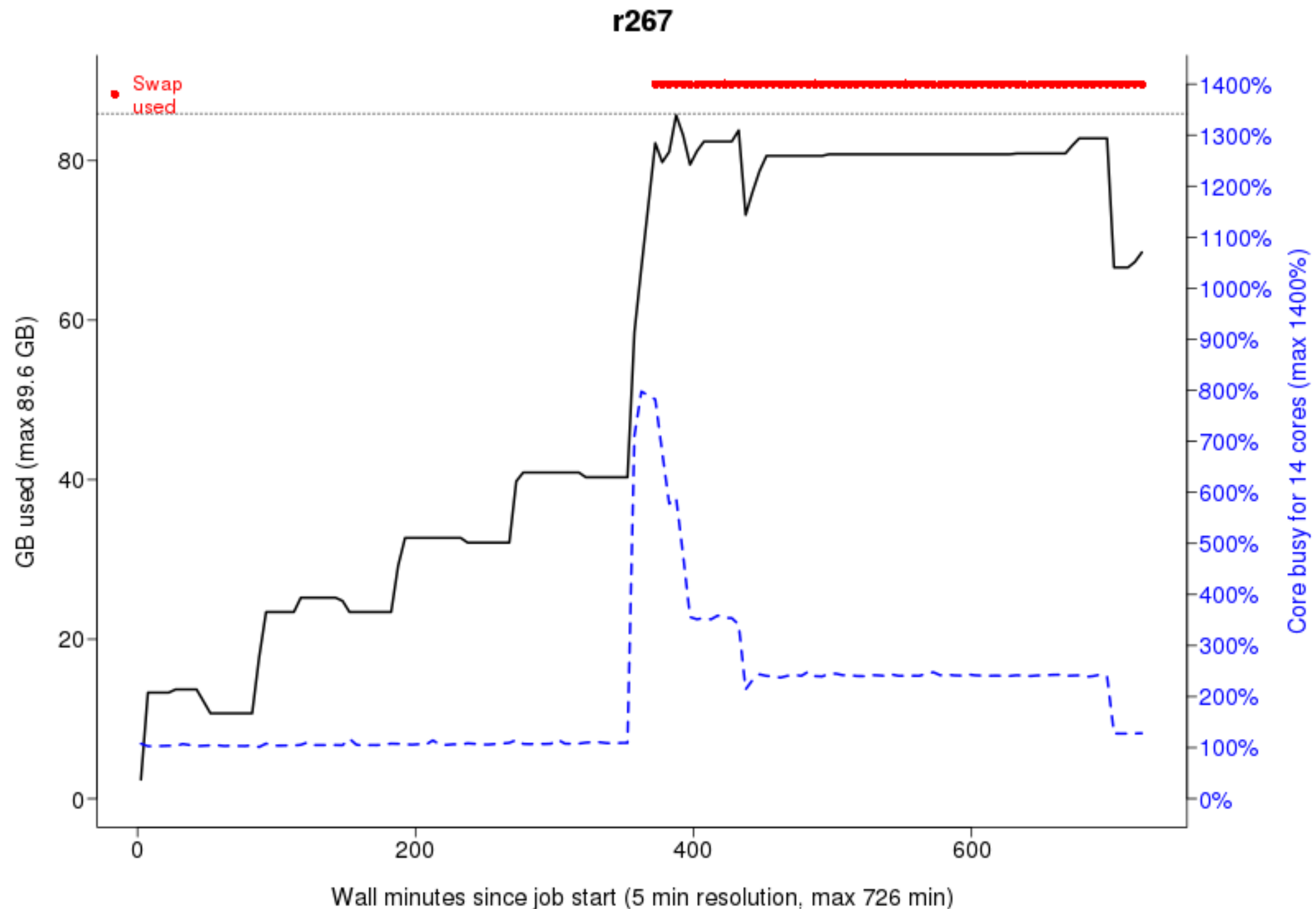
607031 OUT_OF_MEMORY on rackham end: 2018-08-25T07:29:39 runtime: 12:08:5

User:

Proj: snic2017-1-355

Jobname: pgd.apps

!!swap_used, cores_overbooked:14:11





Different flavours of Slurm

Job script examples and workflows



Simple workflow

```
#!/bin/bash
#SBATCH -J jobname
#SBATCH -A sens2023598
#SBATCH -p core
#SBATCH -n 10
#SBATCH -t 10:00:00

module load software/version
module load python/3.11.8

./my-script.sh
./another-script.sh
./myprogram.exe
```



Job dependencies

- `sbatch jobscrip.sh` submitted job with *jobid1*
- `sbatch anotherjobscrip.sh` submitted job with *jobid2*
- `--dependency=afterok:jobid1:jobid2` job will only start running after the successful end of jobs *jobid1:jobid2*
- very handy for clearly defined workflows
- One may also use `--dependency=afternotok:jobid` in case you'd like to resubmit a failed job, OOM for example, to a node with a higher memory: `-C mem215GB` or `-C mem512GB`



I/O intensive jobs: \$SNIC_TMP

```
#!/bin/bash
```

```
#SBATCH -J jobname
```

```
#SBATCH -A sens2023598
```

```
#SBATCH -p core
```

```
#SBATCH -n 1
```

```
#SBATCH -t 10:00:00
```

```
module load bioinfotools
```

```
module load bwa/0.7.17 samtools/1.14
```

```
export SRCDIR=$HOME/path-to-input
```

```
cp $SRCDIR/foo.pl $SRCDIR/bar.txt $SNIC_TMP/.
```

```
cd $SNIC_TMP
```

```
./foo.pl bar.txt
```

```
cp *.out $SRCDIR/path-to-output/.
```



OpenMP or multi-threaded job

```
#!/bin/bash
#SBATCH -A sens2023598
#SBATCH --exclusive
#SBATCH -p node
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=20
#SBATCH -t 01:00:00
```

```
module load uppasd
```

```
export OMP_NUM_THREADS=20
```

```
sd > out.log
```



GPU nodes on Bianca

- Nodes with Nvidia A100 40 GB
- All GPU nodes have at least 256 GB RAM (fat nodes) with 16 CPU cores with 2 GPUs per node

- SBATCH options:

#SBATCH -C gpu

#SBATCH --gpus=2 #number of GPUs requested

#SBATCH --gpus-per-node=2 #number of GPUs per node

nvidia-smi

- https://slurm.schedmd.com/gres.html#Running_Jobs



Running on several nodes: MPI jobs

```
#!/bin/bash -l
#SBATCH -J rsptjob
#SBATCH --mail-type=FAIL
#SBATCH -A sens2023598
#SBATCH -t 00-07:00:00
#SBATCH -p node
#SBATCH -N 4
### for jobs shorter than 15 min (max 4 nodes):
###SBATCH --qos=short
```

```
module load RSPT/2021-10-04
export RSPT_SCRATCH=$SNIC_TMP
```

```
srun -n 80 rspt
```

```
rm -f apts dmft_lock_file e_entropy efgArray.dat.0 efgData.out.0
energy_matrices eparm_last interstitialenergy jacob1 jacob2 locust.*
out_last pot_last rspt_fft_wisdom.* runs.a symcof_new
```



Job arrays

- Submit many jobs at once with the same or similar parameters
- Use `$SLURM_ARRAY_TASK_ID` in the script in order to find the correct path

```
#!/bin/bash
#SBATCH -A sens2023598
#SBATCH -p node
#SBATCH -N 2
#SBATCH -t 01:00:00
#SBATCH -J jobarray
#SBATCH --array=0-19
#SBATCH --mail-type=ALL,ARRAY_TASKS

# SLURM_ARRAY_TASK_ID tells the script which iteration to run
echo $SLURM_ARRAY_TASK_ID

cd /pathtomydirectory/dir_${SLURM_ARRAY_TASK_ID}/

srun -n 40 my-program
env
```

- You may use `scontrol` to modify some of the job arrays.



Snakemake and Nextflow

- Conceptually similar, but with different flavours
- First define steps, each with an input, an output, and a command that transforms the input into output
- Then just ask for the desired output and the system will handle the rest
- Snakemake hackathon: <https://www.naiss.se/event/online-training-snakemake-hackathon-2024-05/>
- Nextflow training: <https://training.nextflow.io/>



Hands-on #4: make it your own

- use 2 or 3 of the sample job scripts as a starting point for your own job script
- tweak them so that you run something closer to your research; or just feel free to experiment
- paste at least one of the examples in the HackMD
- great if you could add a comment what the job script is about





Where to go from here?

Code documentation

SNIC training newsletter - software-specific training events included

<https://coderefinery.org/workshops/upcoming/>

<https://nbis.se/training/events.html> (bio)

contact UPPMAX support via <https://supr.naiss.se/support/>

