Utrecht University

Taming the Scheduler: Primer and Good Practices for Slurm on HPC Systems

Python for lunch - 11/2025

Laptop vs. HPC cluster

Scale ?

Laptop vs. HPC cluster

Scale ?

iPad 2 has as much compute and memory as the Cray-2 (world's largest machine in 1986)

Does that make it an HPC cluster (albeit old) ?

Laptop vs. HPC cluster

~~Scale ?~~

- Optimized for desktop tasks (video, browsing, …)
- Multi-tasking a wide range of applications
- Owner/user controls of the software stack
- You are free to use all the resources
- …

- Optimized for compute tasks
- Increasingly heterogeneous
- Intricate hardware/software stack, resulting from many constraints
- Resources shared by many stakeholders
- …

Laptop vs. HPC cluster

~~Scale ?~~

- Optimized for desktop tasks (video, browsing, …)
- Multi-tasking a wide range of applications
- Owner/user controls of the software stack
- **You are free to use all the resources**
- …

- Optimized for compute tasks
- Increasingly heterogeneous
- Intricate hardware/software stack, resulting from many constraints
- **Resources shared by many stakeholders**
- …

Enters the job scheduler !

- Job scheduler has two main tasks:
  - Provide a fair share of the resources to all the users
  - Maximize the HPC cluster occupancy (conversely, minimize cluster idle time)

➡ These two aspects control the when and where your job starts !

Here, we will consider only the Slurm[1]* job scheduler, but other schedulers can be found in the HPC clusters wilderness (PBS[2], TORQUE[3], ...)

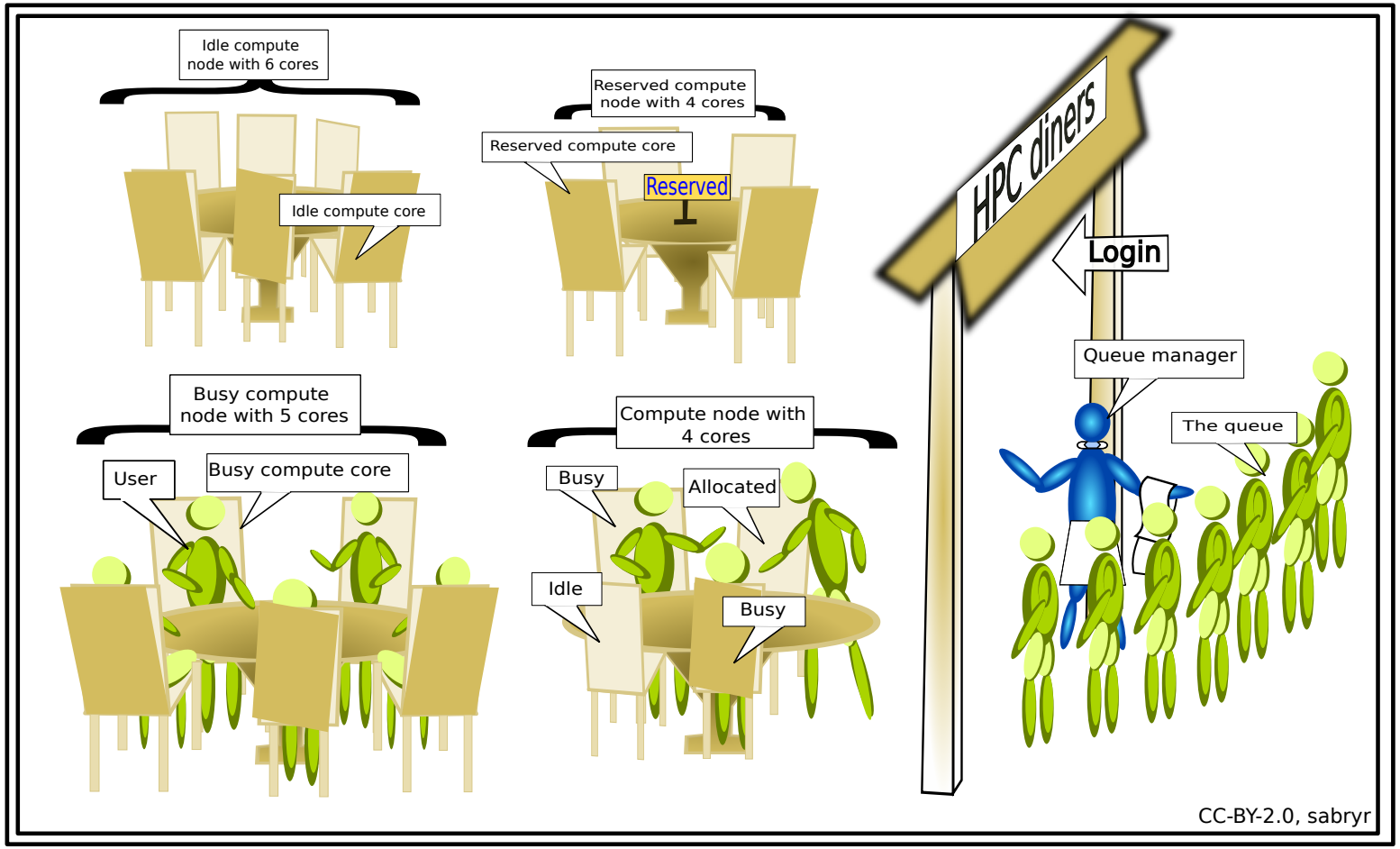
1: https://slurm.schedmd.com/
2: https://ncar-hpc-docs.readthedocs.io/en/latest/pbs/job-scripts/
3: https://hpc-wiki.info/hpc/Torque

* Slurm is an Open Source Software !

Enters the job scheduler !

Each HPC system is different: Slurm will be configured differently on each system !

Enters the job scheduler !



https://carpentries-incubator.github.io/hpc-intro/13-scheduler/index.html

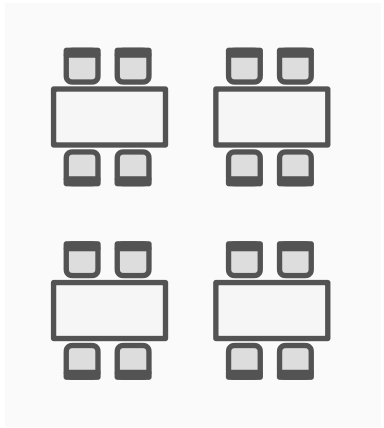
Job scheduler 101

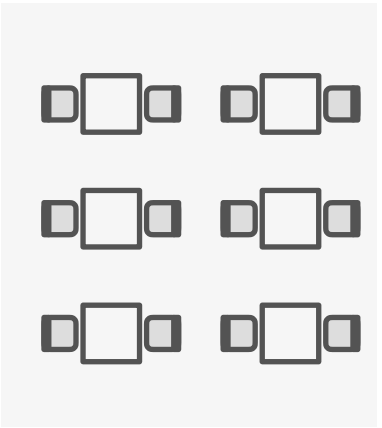
HPC cluster (Snellius)

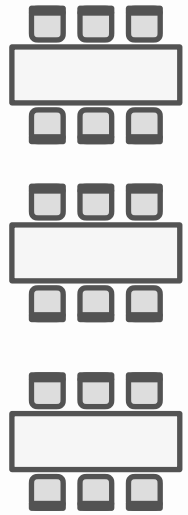
- Various type of **nodes**, depending on the type of hardware on it: *Genoa*, *Rome*, *Milan*, *H100*, …

HPC diner

- Various type of **table groups:**







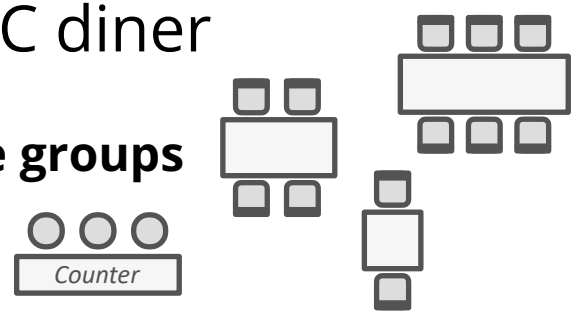
Job scheduler 101

HPC cluster (Snellius)

- Various type of **nodes**, depending on the type of hardware on it: *Genoa*, *Rome*, *Milan*, *H100*, …
- Several **partitions (queues)** are available with different characteristics:
  - Walltime, type of nodes, number of nodes

HPC diner

- Various type of **table groups**



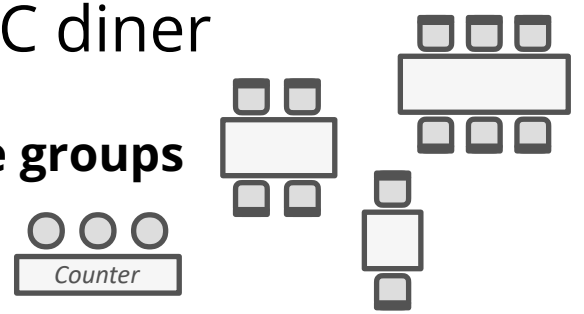
- Several **rooms**, each with a queue. Choose the one depending on your party size, planned duration of your diner, …

Job scheduler 101

HPC cluster (Snellius)

- Various type of **nodes**, depending on the type of hardware on it: *Genoa*, *Rome*, *Milan*, *H100*, …
- Several **partitions (queues)** are available with different characteristics:
  - Walltime, type of nodes, number of nodes
- Billing made automatically to a registered **account**, depending on size and (actual) duration of the job

HPC diner



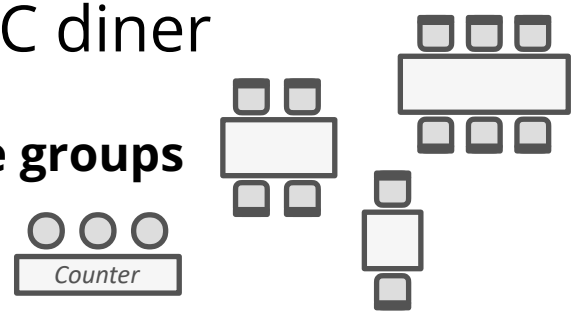
- Various type of **table groups**
- Several **rooms**, each with a queue. Choose the one depending on your party size, planned duration of your diner, …
- Only **known VIP** can book, billed directly to an open tab (with a determined limit of credit)

Job scheduler 101

HPC cluster (Snellius)

- Various type of **nodes**, depending on the type of hardware on it: *Genoa*, *Rome*, *Milan*, *H100*, …
- Several **partitions (queues)** are available with different characteristics:
  - Walltime, type of nodes, number of nodes
- Billing made automatically to a registered **account**, depending on size and (actual) duration of the job
- Node usage can be **exclusive** or **shared**

HPC diner



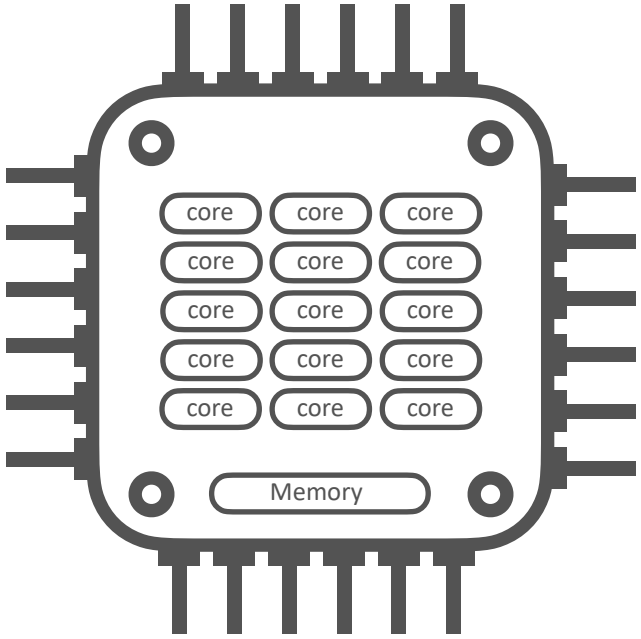
- Various type of **table groups**
- Several **rooms**, each with a queue. Choose the one depending on your party size, planned duration of your diner, …

- Only **known VIP** can book, billed directly to an open tab (with a determined limit of credit)
- You can book an entire (2/4/6 chairs) table group to yourself, or allow others to join

Job scheduler 101

HPC cluster (Snellius)

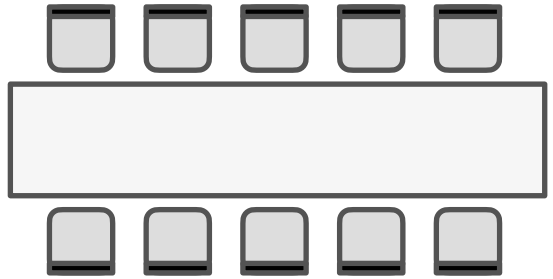
- Modern CPUs are **multicore**:



AMD Genoa CPUs: 96 cores

HPC diner

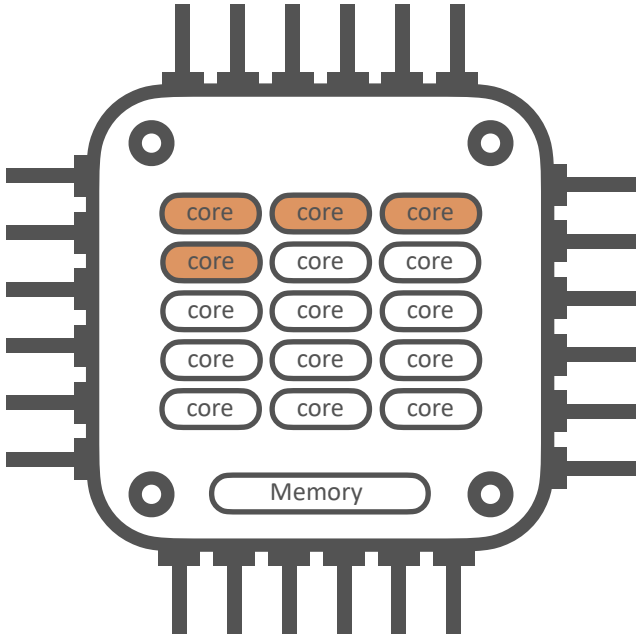
- Tables have a lot of **seats**:



Job scheduler 101

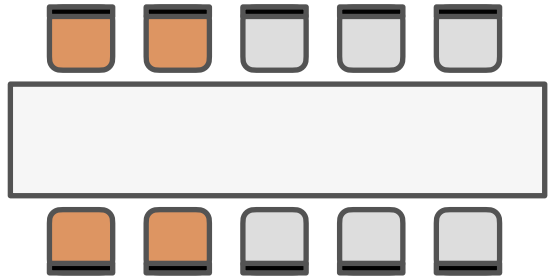
HPC cluster (Snellius)

- Modern CPUs are **multicore**:



HPC diner

- Tables have a lot of **seats**:



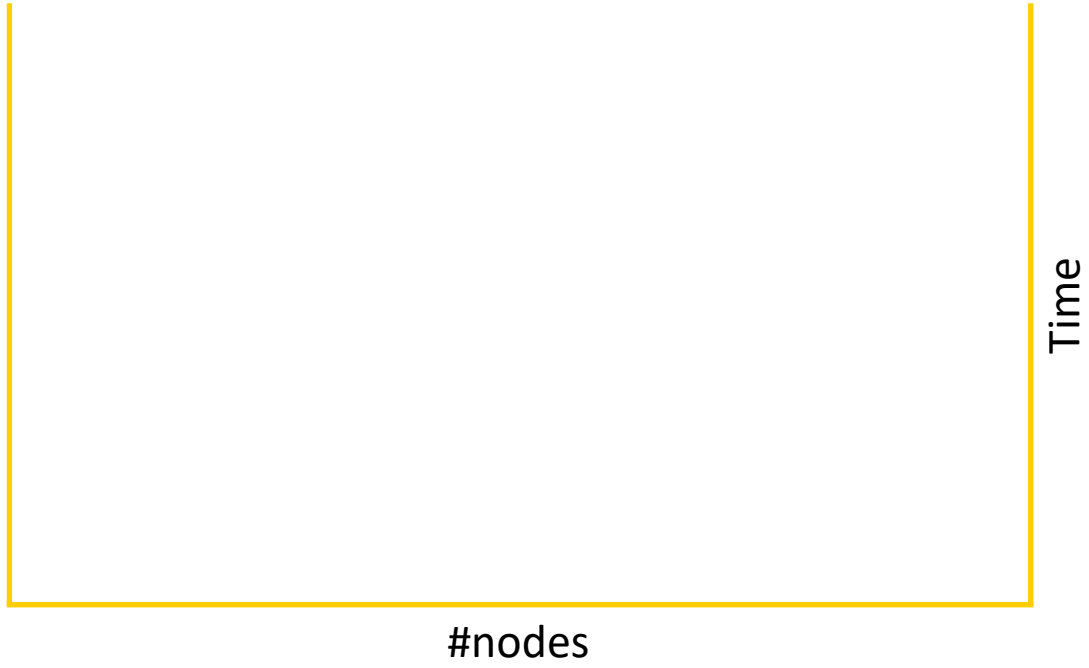
You can choose to only reserve some of the cores (seats) on a table group

Job scheduler 101

What does the job scheduler do in the background ?

- Solving a Tetris problem in a #nodes / time space:

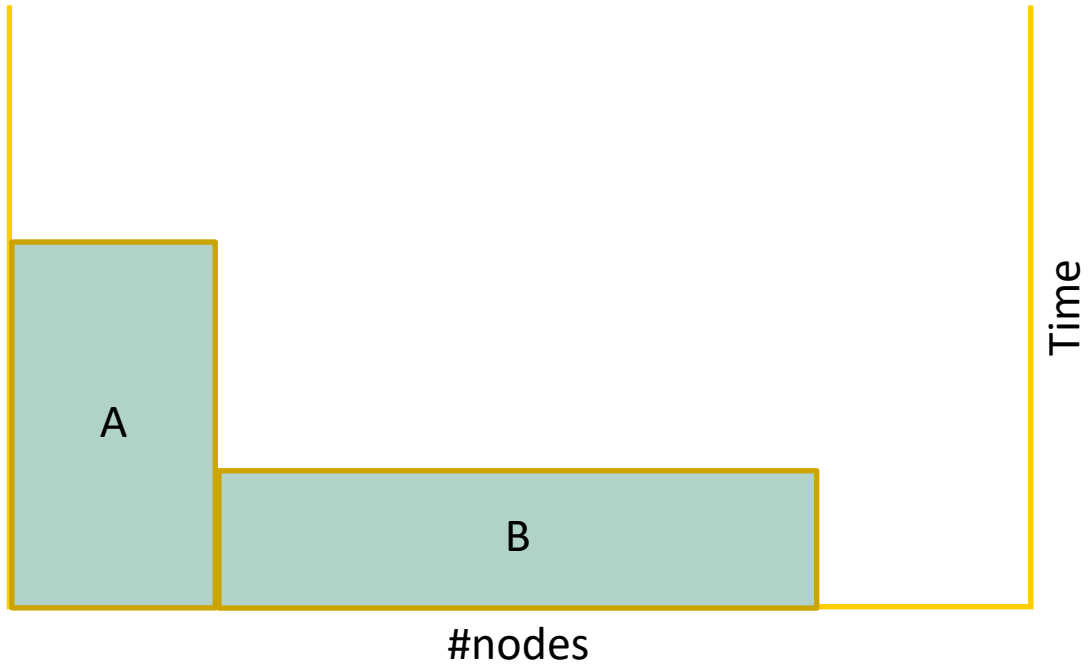
https://servicedesk.surf.nl/wiki/spaces/WIKI/pages/30660219/The+job+scheduler



Job scheduler 101

What does the job scheduler do in the background ?

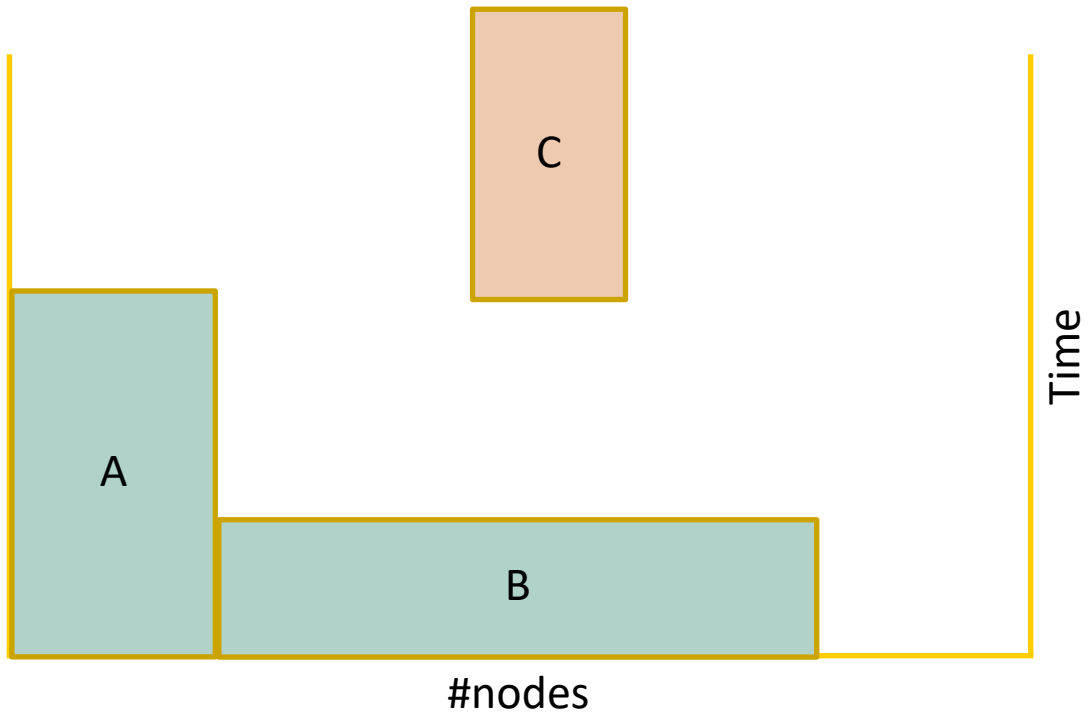
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

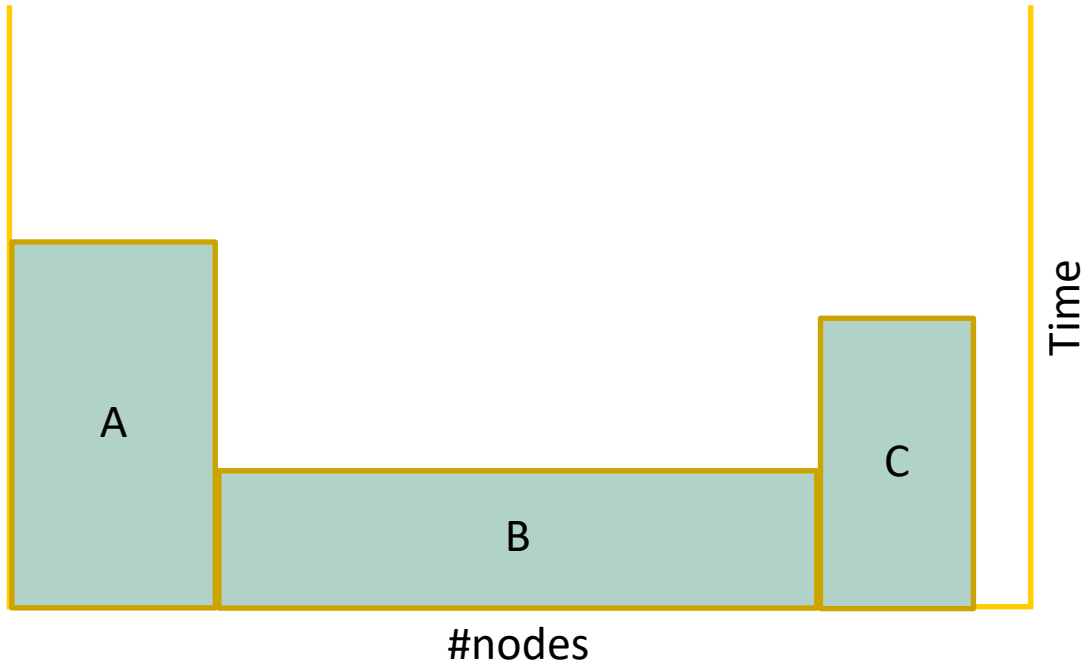
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

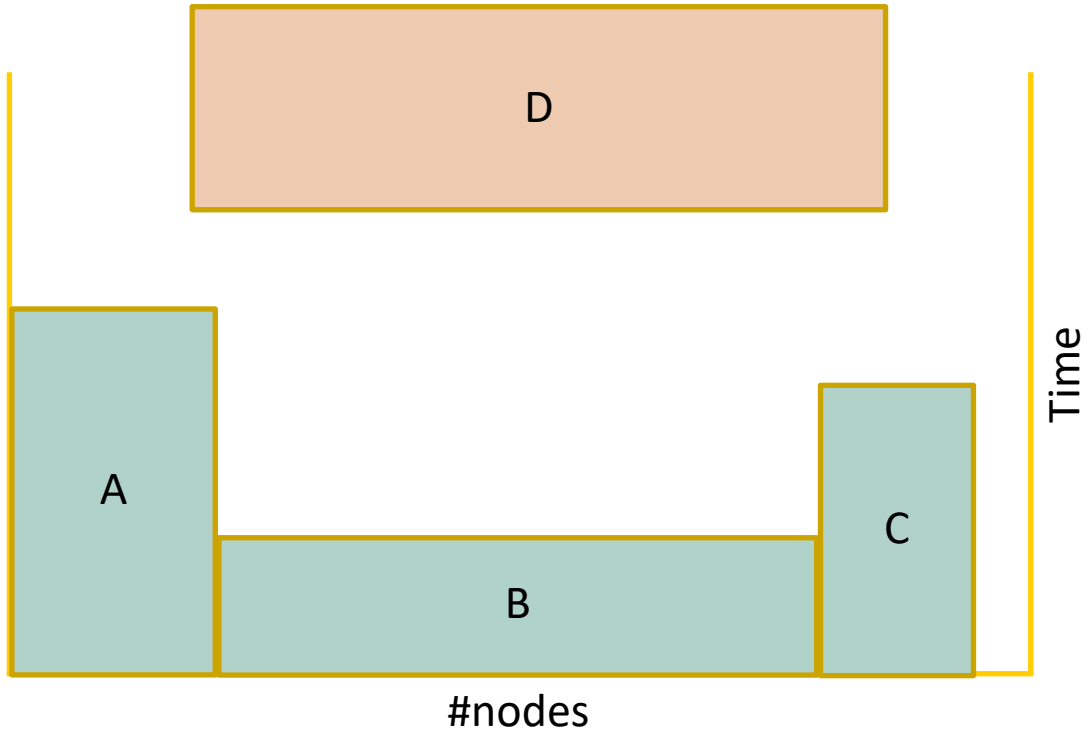
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

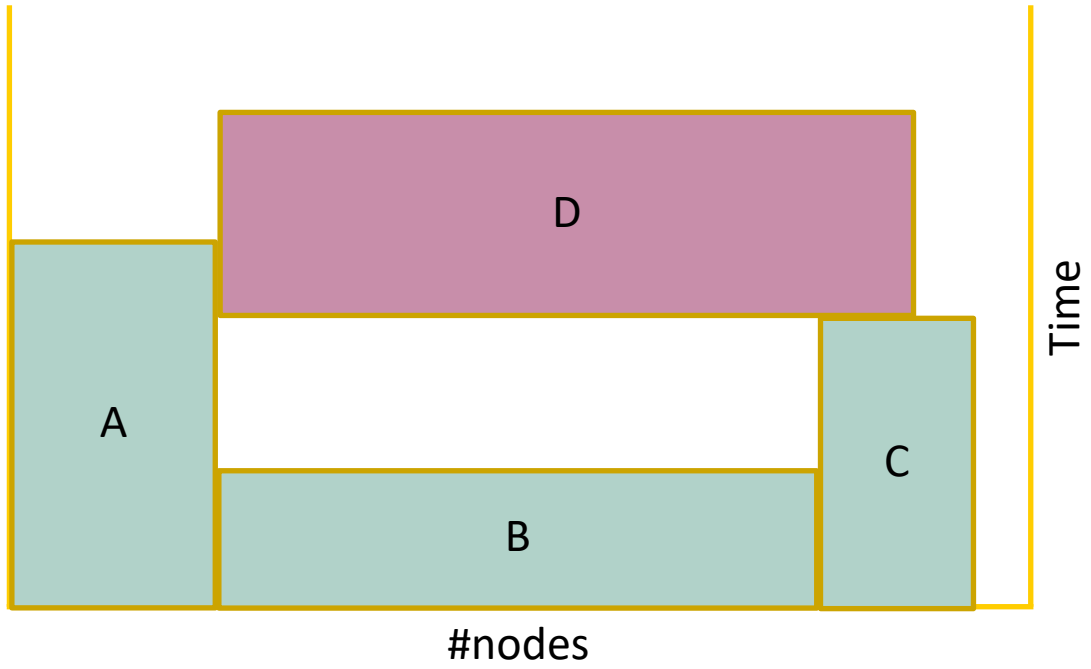
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

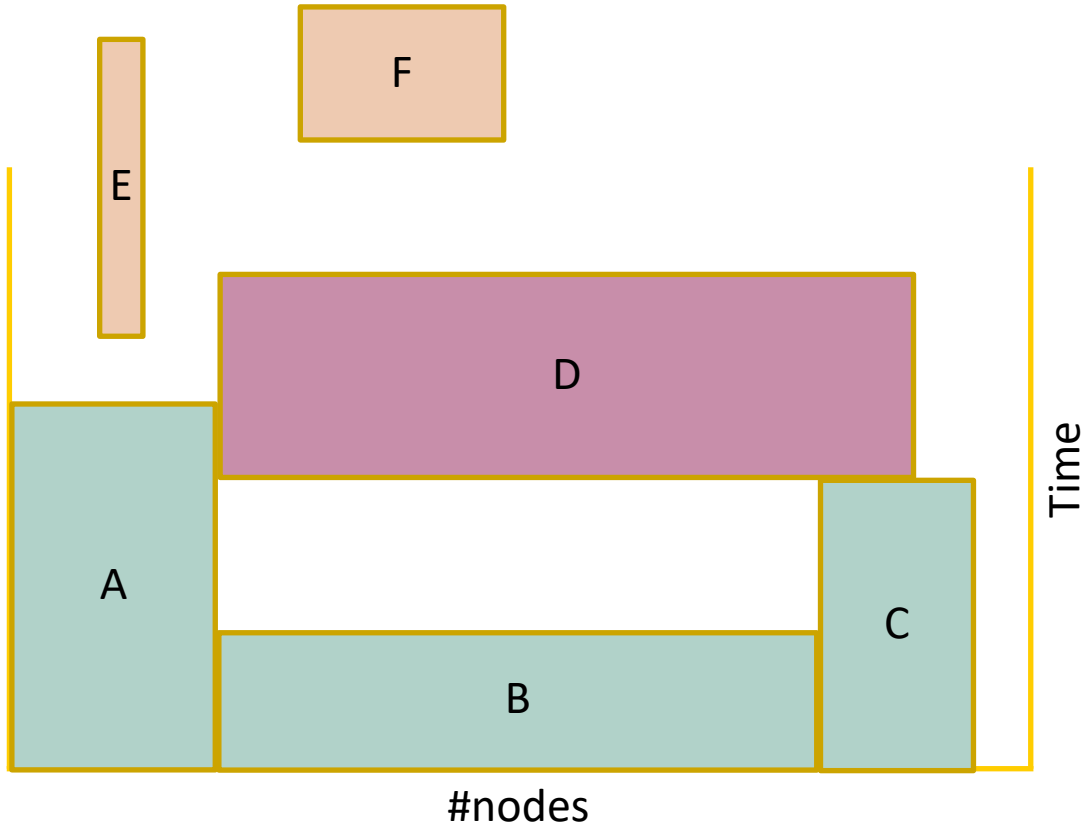
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

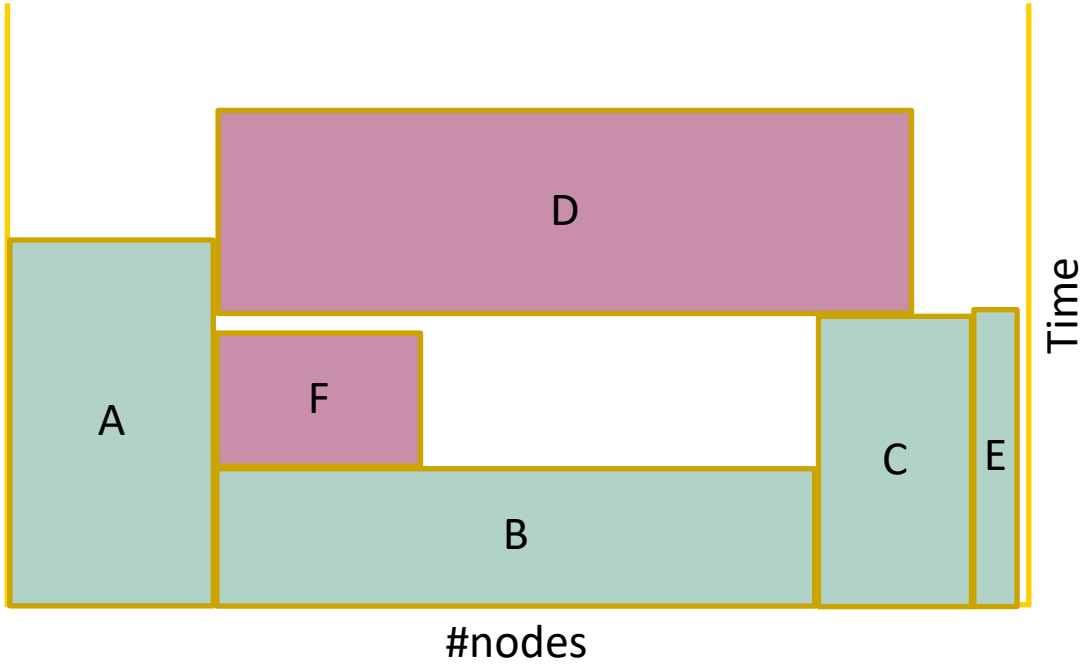
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

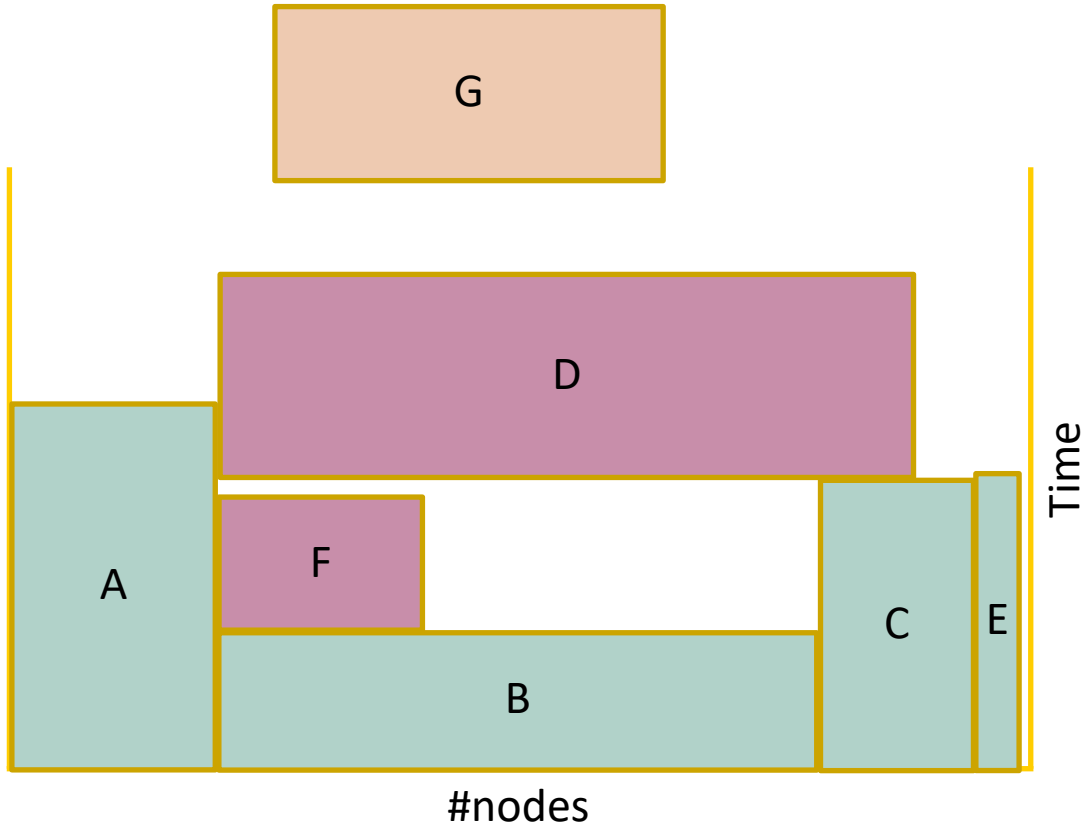
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

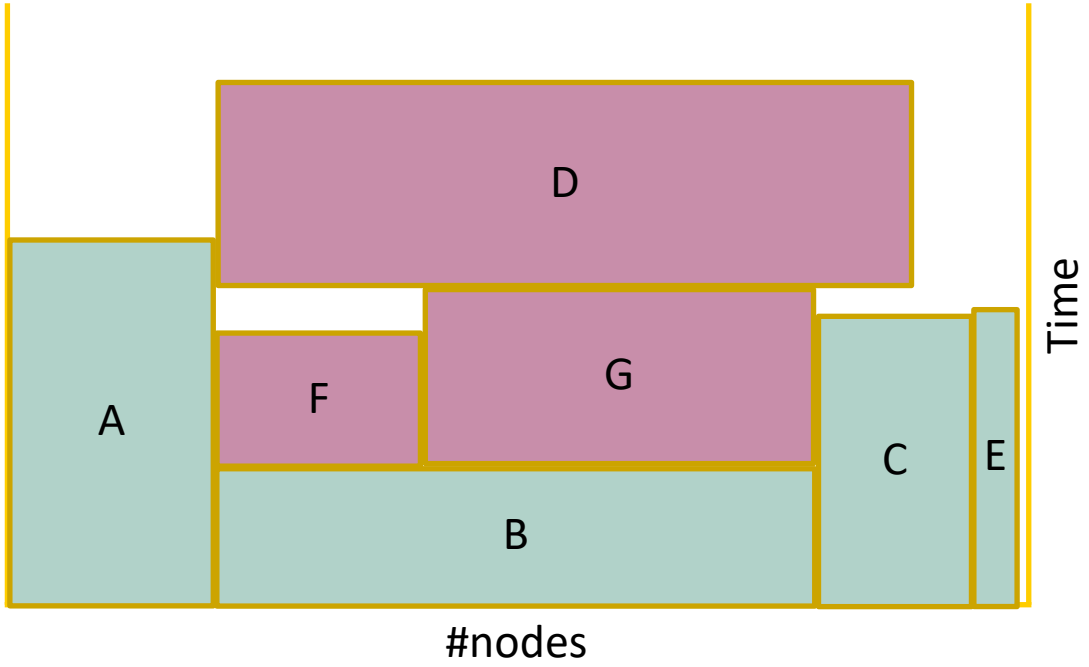
- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

- Solving a Tetris problem in a #nodes / time space:



Job scheduler 101

What does the job scheduler do in the background ?

- Solving a Tetris problem in a #nodes / time space: the objective is to maximize occupancy
- Occupancy is not the only constrain: each job has a *priority* based on job age, user fair share, job size, …
- This is actually a difficult optimization problem, that is solved regularly as jobs are added/removed from each queue

Gather information on your cluster

Read the documentation of your cluster !

- On most public-accessed clusters, there is extensive documentation about the available hardware, software, login procedures, …
- Often the doc contains a primer on Slurm and specific Slurm settings that your system administrator has set up

https://servicedesk.surf.nl/wiki/spaces/WIKI/pages/30660184/Snellius

Gather information on your cluster

List available resources, partitions, …

- `sinfo -s`: a condensed summary of the available partition

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
rome*        up 5-00:00:00    294/224/4/522 tcn[4-525]
rome_fnwi    up 5-00:00:00       12/0/0/12 tcn[4-15]
genoa        up 5-00:00:00    452/283/2/737 tcn[527-1263]
fat_rome     up 5-00:00:00       19/51/2/72 fcn[1-72]
fat_genoa    up 5-00:00:00        47/1/0/48 fcn[73-120]
gpu_a100     up 5-00:00:00        62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
gpu_h100     up 5-00:00:00       66/21/1/88 gcn[73-160]
gpu_mig      up 5-00:00:00          3/1/0/4 gcn[2-5]
gpu_vis      up 1-00:00:00        62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
himem_4tb    up 5-00:00:00          0/2/0/2 hcn[1-2]
himem_8tb    up 5-00:00:00          1/1/0/2 hcn[3-4]
```

Gather information on your cluster

List available resources, partitions, …

- `sinfo -s`: a condensed summary of the available partition

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
rome*        up 5-00:00:00    294/224/4/522 tcn[4-525]
rome_fnwi    up 5-00:00:00       12/0/0/12 tcn[4-15]
genoa        up 5-00:00:00    452/283/2/737 tcn[527-1263]
fat_rome     up 5-00:00:00       19/51/2/72 fcn[1-72]
fat_genoa    up 5-00:00:00        47/1/0/48 fcn[73-120]
gpu_a100     up 5-00:00:00        62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
gpu_h100     up 5-00:00:00       66/21/1/88 gcn[73-160]
gpu_mig      up 5-00:00:00          3/1/0/4 gcn[2-5]
gpu_vis      up 1-00:00:00        62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
himem_4tb    up 5-00:00:00          0/2/0/2 hcn[1-2]
himem_8tb    up 5-00:00:00          1/1/0/2 hcn[3-4]
```

Partition name, is it available to you ?

Gather information on your cluster

List available resources, partitions, …

- `sinfo -s`: a condensed summary of the available partition

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
rome*        up 5-00:00:00   294/224/4/522 tcn[4-525]
rome_fnwi    up 5-00:00:00      12/0/0/12 tcn[4-15]
genoa        up 5-00:00:00   452/283/2/737 tcn[527-1263]
fat_rome     up 5-00:00:00     19/51/2/72 fcn[1-72]
fat_genoa    up 5-00:00:00      47/1/0/48 fcn[73-120]
gpu_a100     up 5-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
gpu_h100     up 5-00:00:00     66/21/1/88 gcn[73-160]
gpu_mig      up 5-00:00:00        3/1/0/4 gcn[2-5]
gpu_vis      up 1-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
himem_4tb    up 5-00:00:00        0/2/0/2 hcn[1-2]
himem_8tb    up 5-00:00:00        1/1/0/2 hcn[3-4]
```

Partition name, is it available to you ?

Walltime limit

Gather information on your cluster

List available resources, partitions, …

- `sinfo -s`: a condensed summary of the available partitions and nodes

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
rome*        up 5-00:00:00   294/224/4/522 tcn[4-525]
rome_fnwi    up 5-00:00:00      12/0/0/12 tcn[4-15]
genoa        up 5-00:00:00   452/283/2/737 tcn[527-1263]
fat_rome     up 5-00:00:00     19/51/2/72 fcn[1-72]
fat_genoa    up 5-00:00:00      47/1/0/48 fcn[73-120]
gpu_a100     up 5-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
gpu_h100     up 5-00:00:00     66/21/1/88 gcn[73-160]
gpu_mig      up 5-00:00:00        3/1/0/4 gcn[2-5]
gpu_vis      up 1-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
himem_4tb    up 5-00:00:00        0/2/0/2 hcn[1-2]
himem_8tb    up 5-00:00:00        1/1/0/2 hcn[3-4]
```

Partition name, is it available to you ?

Walltime limit

Number of nodes:
***A**llocated, **I**dle, **O**ut, **T**otal*

Gather information on your cluster

List available resources, partitions, …

- `sinfo -s`: a condensed summary of the available partitions and nodes

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
rome*        up 5-00:00:00   294/224/4/522 tcn[4-525]
rome_fnwi    up 5-00:00:00      12/0/0/12 tcn[4-15]
genoa        up 5-00:00:00   452/283/2/737 tcn[527-1263]
fat_rome     up 5-00:00:00     19/51/2/72 fcn[1-72]
fat_genoa    up 5-00:00:00      47/1/0/48 fcn[73-120]
gpu_a100     up 5-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
gpu_h100     up 5-00:00:00      66/21/1/88 gcn[73-160]
gpu_mig      up 5-00:00:00        3/1/0/4 gcn[2-5]
gpu_vis      up 1-00:00:00      62/0/1/63 gcn[6-8,10-49,51,53-62,64-72]
himem_4tb    up 5-00:00:00        0/2/0/2 hcn[1-2]
himem_8tb    up 5-00:00:00        1/1/0/2 hcn[3-4]
```

Need to run on GPU right now ? No idle A100 nodes, use H100 instead !

Partition name, is it available to you ?

Walltime limit

Number of nodes: ***A****llocated,* ***I****dle,* ***O****ut,* ***T****otal*

Interacting with the scheduler: interactive job

Run an interactive job for a small task:

- `srun <myscript>`
  - Launching a small script that maybe still requires a lot of memory/CPU. You will also enters the Slurm queue and your terminal will hang until the job start.
  - Note: by default the queue listed with a `*` in `sinfo -s` will be used
- `srun -p genoa -t 1:00:00 -pty /bin/bash`
  - Launching a job and logging on the compute node directly. Then you can run multiple script or debug a workflow.
- Interactive jobs are a great way to test your workflow, benchmark the performance to better dimension your *production* jobs !

Interacting with the scheduler: job script

Write a job script:

- Header: Slurm commands are prepended with the `#SBATCH` and set the parameters mentioned earlier (see HPC diner example): **node**, **partition**, **account**, …

```
#!/bin/bash
#SBATCH -p genoa
#SBATCH –a <mypropject>
#SBATCH -j <jobname>
#SBATCH -t 12:00:00
#SBATCH -N 1
#SBATCH -n 128
```

- ➡ Partition
- ➡ Account
- ➡ Job name
- ➡ Wall clock time
- ➡ Number of nodes
- ➡ Number of tasks

Interacting with the scheduler: job script

Write a job script:

- Header
- Body of the script: load modules, set your environment and launch your application
- The script (and the job) will end when all the *job steps* are finished (or one fails)

```
#!/bin/bash
#SBATCH -p genoa
#SBATCH –a <mypropject>
#SBATCH -j <jobname>
#SBATCH -t 12:00:00
#SBATCH -N 1
#SBATCH -n 128
```

- ➡ Partition
- ➡ Account
- ➡ Job name
- ➡ Wall clock time
- ➡ Number of nodes
- ➡ Number of tasks

```
module load 2025 foss/2025a Python CCSM

srun -n1 preprocess_ccsm.exe >& ccsm_pre.log
srun ./ccsm.exe >& ccsm.log
srun -n1 postprocess_ccsm.exe >& ccsm_post.log
```

Interacting with the scheduler: job script

Submit your job script to Slurm queue:

- `sbatch <myjobscript>`

- Slurm will add your job to the queue specified in the script and associate a job id:

```
Submitted batch job 15961002
```

- If something is wrong with your script, Slurm will let you know right away:

```
sbatch: error: Batch job submission failed: Requested time limit is invalid (missing or exceeds some limit)
```

Interacting with the scheduler: job script

Submit your job script to Slurm queue:

- `sbatch <myjobscript>`

- Most of the Slurm parameters can be directly specified from the command line (taking precedence):
  - `sbatch -p genoa -t 1-00:00:00 <myjobscript>`

Interacting with the scheduler: job script

Monitor and control your job:

- `squeue -u <username>`

```
   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
15961002     genoa batch.SN lesclape PD       0:00      1 (None)
```

Interacting with the scheduler: job script

Monitor and control your job:

- `squeue -u <username>`

```
   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
15961002     genoa batch.SN lesclape PD       0:00      1 (None)
```

Job status:
- PD: pending
- R: running
- S: Suspended
- CG: Completing
- CD: Completed
- F: Failed

Reason for holding the job:
- Priority
- Dependency
- Resources

Interacting with the scheduler: job script

Monitor and control your job:

- `squeue -u <username>`

```
   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
15961002     genoa batch.SN lesclape PD       0:00      1 (None)
```

- `scancel <jobid>`
- `sacct -u lesclapez01`

```
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
15961002     batch.SNE+      genoa   ncuuh284          0 CANCELLED+      0:0
```

Interacting with the scheduler: job script

Write a job script: more advanced generic parameters

- Many Slurm parameters can be specified in the header, some useful ones:
  - `--exclusive`: to request exclusive access to a node (default is often shared access)
  - `--dependency=<state:jobid>`: introduce dependencies between jobs, the state can be one of `after`, `afterany`, `afterok`, `afternotok`
  - `--mem-per-cpu=<MB>`: to specify the memory per CPU
  - `--mail-user=<youremail@domain.com>`
    `--mail-type=<BEGIN,END,FAIL>`: to receive email when the job starts, fails or ends
  - `--output=logs/%x_%j.out`: to control the standard output (%x and %j are neat shortcuts for `SLURM_JOB_NAME` and `SLURM_JOB_ID`

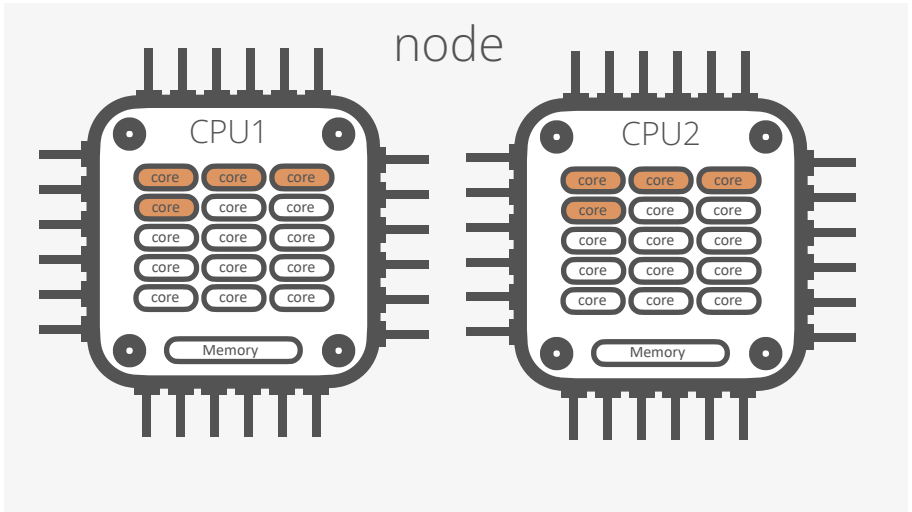
https://slurm.schedmd.com/pdfs/summary.pdf

Interacting with the scheduler: Slurm compute environment

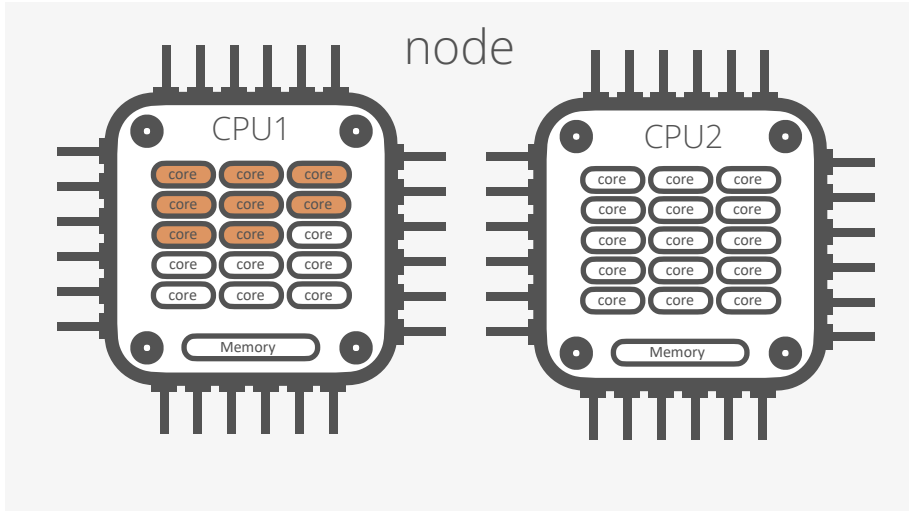
- Upon starting, the content of a job script is launched in a fresh shell:
  - ➡ Don't forget to load your compute environment (module, environment variables, …) !
- Slurm create a few handy environment variables:
  - `SLURM_JOB_ID`, `SLURM_JOB_NAME`, `SLURM_SUBMIT_DIR`, `SLURM_JOB_START_TIME`, `SLURM_JOB_TIME_LIMIT`, …
- When launching an MPI application, it is advised to use `srun` instead of bare `mpirun` to let Slurm control set the number of tasks, control task/core mapping, CPU-GPU binding, …

Interacting with the scheduler: Slurm compute environment

- Upon starting, the content of a job script is launched in a fresh shell:
-
-



Task to core mapping with *mpirun*



Task to core mapping with *srun*

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)

```
#SBATCH -j serial
#SBATCH -n 1
#SBATCH --cpus-per-task=1
#SBATCH -t 00:30:00
#SBATCH --mem=2G

srun ./my_program
```

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)
  - Single process, multiple threads (8 cores)

```
#SBATCH -j serial
#SBATCH -n 1
#SBATCH --cpus-per-task=8
#SBATCH -t 00:30:00
#SBATCH --mem=2G

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun ./my_threaded_program
```

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)
  - Single process, multiple threads (8 cores)
  - Multiple processes, single threads (8 cores) on a single node

```
#SBATCH -j serial
#SBATCH -n 8
#SBATCH --cpus-per-task=1
#SBATCH -t 00:30:00
#SBATCH --mem=2G

srun ./my_mpi_program
```

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)
  - Single process, multiple threads (8 cores)
  - Multiple processes, single threads (8 cores) on a single node
  - Multiple processes, single threads (32 cores) on two nodes

```
#SBATCH -j serial
#SBATCH -N 2
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=1
#SBATCH -t 00:30:00
#SBATCH –mem-per-cpu=2G

srun ./my_mpi_program
```

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)
  - Single process, multiple threads (8 cores)
  - Multiple processes, single threads (8 cores) on a single node
  - Multiple processes, single threads (32 cores) on two nodes
  - Multiple processes, multiple threads (32 cores) on two nodes

```
#SBATCH -j serial
#SBATCH -N 2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=4
#SBATCH -t 00:30:00
#SBATCH --mem=8G

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun ./my_hybrid_program
```

Interacting with the scheduler: Slurm compute environment

- Requesting resources: handling processes/threads
  - Single process, single thread (1 core)
  - Single process, multiple threads (8 cores)
  - Multiple processes, single threads (8 cores) on a single node
  - Multiple processes, single threads (32 cores) on two nodes
  - Multiple processes, multiple threads (32 cores) on two nodes

- Can get more complicated when using GPUs, heterogeneous jobs (mixing different partitions), …

Interacting with the scheduler: Slurm compute environment

- Starting a Slurm job script takes time (from a few seconds to minutes if your job requires thousands of nodes)
- If you have many small similar jobs to launch:
  - use Slurm job arrays `sbatch --array=1-10 <myscript>`: 10 independent jobs, each with his own ID will be submitted. Utilize `SLURM_ARRAY_TASK_ID` to distinguish what each job is doing (different datasets, …).
  - Within a job script, launch multiple executable within the script (aka *job steps*):
    - Reduce Slurm overhead
    - Beware of not oversubscribing the resources

```
for f in input{1..4}.dat; do
    srun -n1 ./analyze $f &
done
wait
```

Interacting with the scheduler: General tips

- Know your environment !
  - Read the cluster documentation, use `sinfo -s`, reach out to the system admin if something is not clear or missing
  - Select a partition with more **Idle** nodes if it is available to you
  - It is never a good idea to *"just run on the login node, it's a small script …"*, use bare `srun` to start an interactive session

Interacting with the scheduler: General tips

- Know your environment !
- Know your problem !
  - Test your workflow/run on a smaller/shorter version of the *production* case, possibly using an interactive session
  - If you plan on using many cores, perform a small scaling test (*strong scaling*: keep your problem size fixed, increase the number of cores from 1 to 2x your initial number of cores)
  - Dimension your job to your needs, with a small safety margin for overhead (~10 %). Otherwise you might stay in queue for longer.

Interacting with the scheduler: General tips

- Know your environment !
- Know your problem !
- Make your life easier with shell aliases/environment variables:
  - Job scripts are Shell scripts, so if your workflow involve complex environment setups, add handy functions to your `.bashrc`/`.zshrc` to reduce the risk of errors

```
boot_eTAOC_2025 () {
  module purge
  module load 2025
  module load foss/2025a
  module load netCDF/4.9.3-gompi-2025a
  module load netCDF-Fortran/4.6.2-gompi-2025a
  module load Python/3.13.1-GCCcore-14.2.0
  module load CMake/3.31.3-GCCcore-14.2.0
  module load OpenMPI/5.0.7-GCC-14.2.0
}
```

Interacting with the scheduler: General tips

- Know your environment !
- Know your problem !
- Make your life easier with shell aliases/environment variables:
  - Job scripts are Shell scripts, so if your workflow involves complex environment setups, add handy functions to your `.bashrc`/`.zshrc` to reduce the risk of errors
  - Job scripts are part of your workflow, do not hesitate to add them to your Git repository !

Interacting with the scheduler: General tips

- Know your environment !
- Know your problem !
- Make your life easier with shell aliases/environment variables
- Be patient ;) !
  - If your job has been in queue for a while, it's nothing personal, the scheduler is doing his job and it might be a busy time of the year (before Christmas ??).
  - If you have decided to use an HPC cluster, you probable could not have run the workflow on your laptop, so the wait is worth it !

Additional resources

- SchedMD Slurm site: https://slurm.schedmd.com/slurm.html
- Many universities have a Slurm introductions on their HPC cluster pages:
  - https://stanford-rc.github.io/docs-earth/docs/slurm-basics
  - https://support.ceci-hpc.be/doc/SubmittingJobs/SlurmTutorial/#shared-memory-example-openmp
  - https://servicedesk.surf.nl/wiki/spaces/WIKI/pages/30660217/Creating+and+running+jobs

Two pager cheat sheet:

https://slurm.schedmd.com/pdfs/summary.pdf