Lab03 – Slurm job definition

16.10.2024

Previous lab

- Any issues completing the tasks from Lab02?
- Be cautious when copying from .doc documents!

Slurm resource manager/scheduler

- Was an abbreviation of: Simple Linux Utility for Resource Management
- Slurm is responsible for:
    - Resource management
    - Scheduling
    - Interaction with the cluster
        - Not a policing tool
        - No in-depth knowledge of what is being computed
- Links:
    - Information specific for Ares https://docs.cyfronet.pl/display/~plgpawlik/Ares
    - Information specific for Athena https://docs.cyfronet.pl/display/~plgpawlik/Athena
    - Quick start: https://kdm.cyfronet.pl/portal/Podstawy:SLURM
    - Rules: https://kdm.cyfronet.pl/portal/Zeus:Podstawy#Zasady_obowi.C4.85zuj.C4.85ce_na_klastrze_Zeus (rules apply to all Cyfronet's clusters!)
    - Slurm intro: https://slurm.schedmd.com/quickstart.html

Basic commands

- sinfo – info about nodes
- squeue – queue status
- sbatch – submit a job
- srun – **job step (if used inside of a job script)**
- scancel – cancel job
- sacct – accounting information

Resource specification

- Node -> tasks -> cores
- How to specify resources? With -N, -n, --ntasks-per-node etc. man sbatch
- sbatch -N 1 – Node count
- sbatch -N 1 --ntasks-per-node=4 – one node with 4 tasks per node = 4 cores
- sbatch -N 1 --ntasks-per-node=4 --cpus-per-task=12 – one node with 4 tasks, each has 12 cores, 48 in total
- sbatch -N 2 --ntasks-per-node=4 --cpus-per-task=12, same as above but for 2 nodes, 96 cores in total

- **sbatch -A plglscclass24-cpu – declare using the plglscclass24-cpu grant for your computations, this is important if you have other computing grants and want to use a specific one!**
- The account name plglscclass24-cpu can be determined by inspecting the `hpc-grants` output, and looking for “allocation” name. Please note that there can be multiple grants, use the most recent one!
- Be cautious about available node configurations, you are responsible for “fitting in”
- In most cases jobs are specified with “job scripts”
- Example job script (*.sh file), parameters can be included in the job script or be supplied from the command line as sbatch arguments:

Example no. 1:

```
#!/bin/bash -l
#SBATCH --nodes=10
#SBATCH --ntask-per-node=4
#SBATCH --cpus-per-task=12
#SBATCH --account=plgglscclass24
#SBATCH --partition=plgrid
#SBATCH --time=02:00:00

#initialization
module load gromacs/2023-foss-2021b-plumed-2.9b

# data handling and work
cd $SLURM_SUBMIT_DIR
mpiexec gmx_mpi mdrun -noappend -deffnm dyn -plumed ...

# this will result in 10*4=40 app processes each using 12 cores, 480 in total
```

Example no. 2:

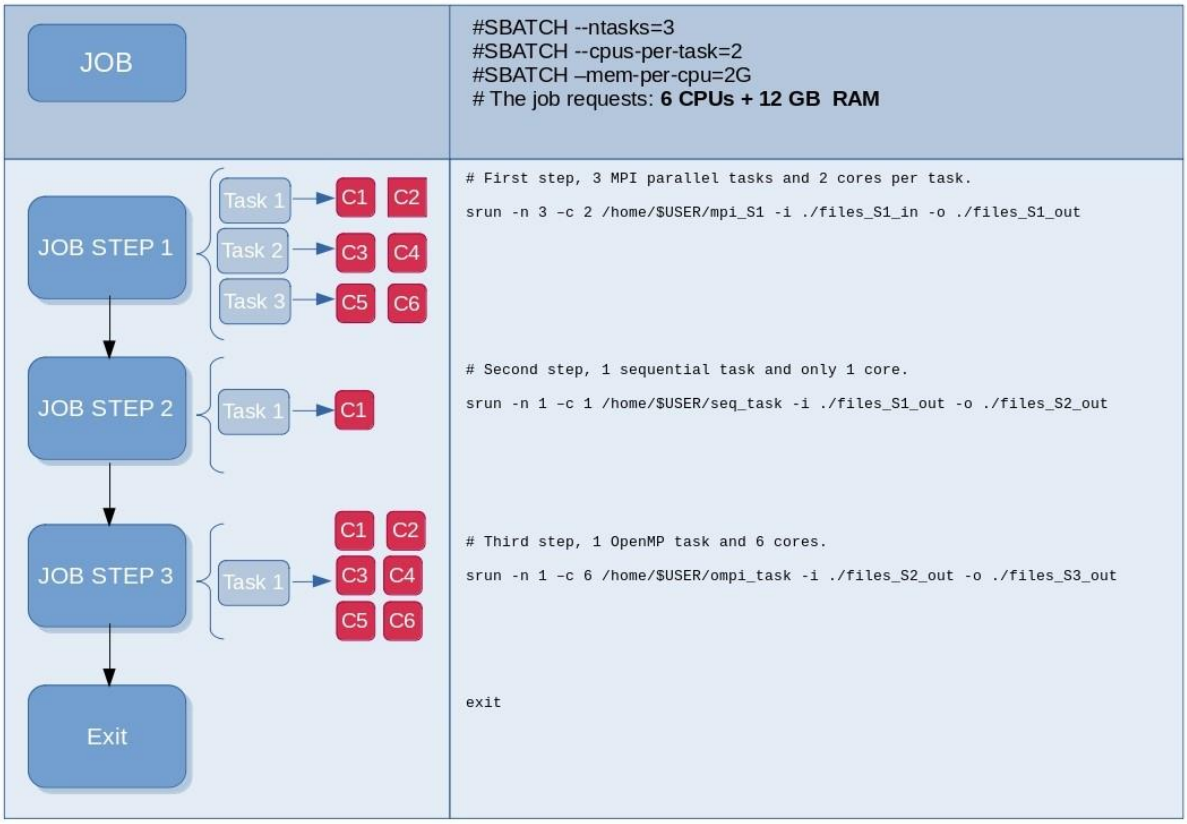

Figure 1: Example SLURM job structure and script. (Source: https://garnatxadoc.uv.es/slurm/slurm_info.html)

Example no. 3:

```
#!/bin/bash -l
#SBATCH --nodes=2
#SBATCH --ntask-per-node=2
#SBATCH --cpus-per-task=24
#SBATCH --time=01:00:00

srun --nodes=1 --ntasks-per-node=1 --cpus-per-task=48 do_preprocessing.bin ...
srun --nodes=2 --ntasks-per-node=2 --cpus-per-task=24 do_computations.bin ...

# end of the script

$ sbatch ...

$ sacct -j 11966005 -o jobid%20,alloctres%40
               JobID                                 AllocTRES
-------------------- ----------------------------------------
            11966005     billing=96,cpu=96,mem=369600M,node=2
      11966005.batch                cpu=48,mem=184800M,node=1
          11966005.0                cpu=48,mem=184800M,node=1
          11966005.1                cpu=96,mem=369600M,node=2
```

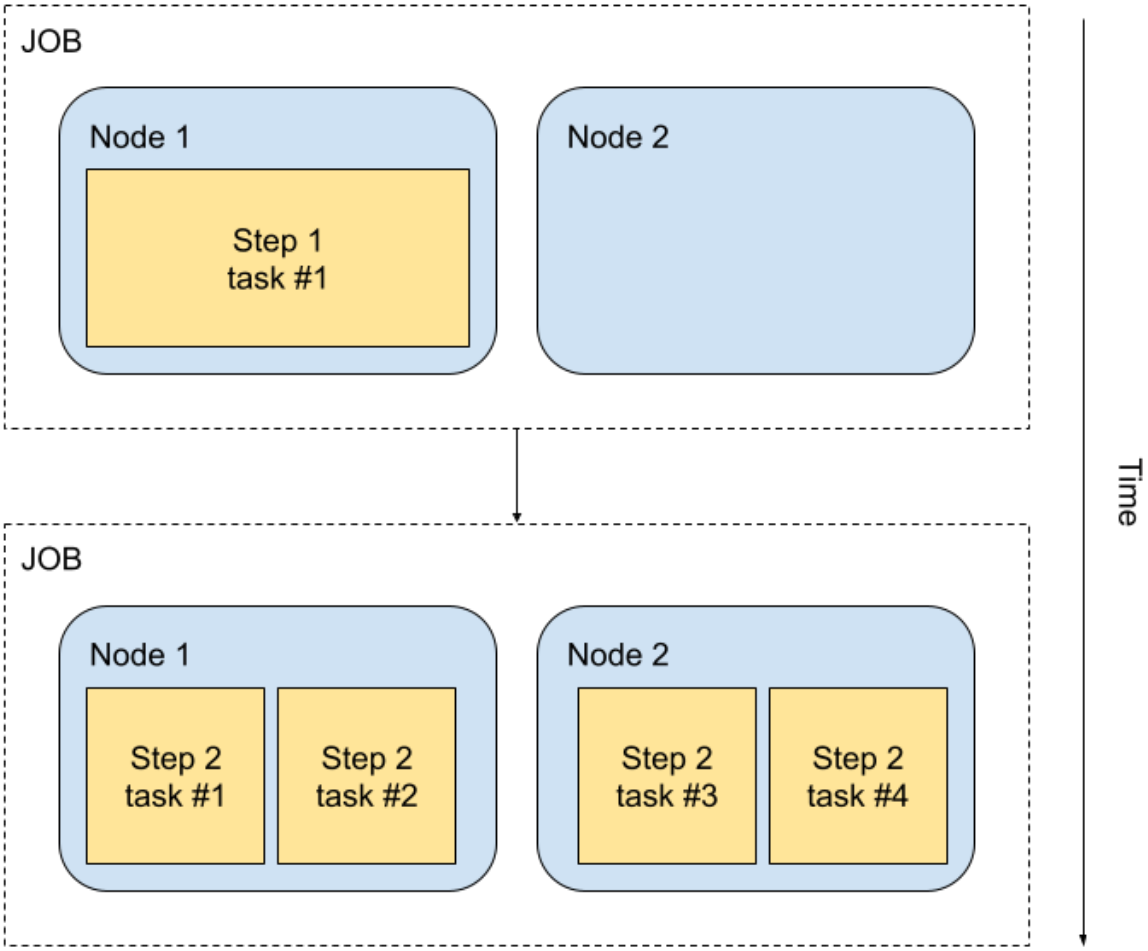

Figure 2: Example SLURM job mapping to resouces over time

- The resources specified by srun... command:
  - can not exceed job's resources!
  - should be available, if not slurm will complain and timeout the request after a while

Environment

- SLURM communicates a lot of information through environment variables
  - Also job configuration and allocated resources
- Other variables: man sbatch, or check with 'env' command in an interactive job

Simple "bag of tasks" workload processing

- The challenge might be "embarrassingly parallel"
- Good examples: image processing, dataset manipulation, etc.
- Each task should be run by srun
- Usefull tools command line tools allowing for parallelization, xargs, parallel, e.g.:
  - echo 1 2 3 | xargs –n1 -P<number of instances> command_to_run <parameter>
  - The same thing can be achieved with "parallel"

- If xargs is given more input than –P, some tasks will wait

xargs demo (sleep sort implementation):

```
$ cat sleeper.sh
#!/bin/bash
sleep $1
echo $1

$ chmod +x sleeper.sh

$ echo 2 1 3 | xargs -t -n1 -P3 ./sleeper.sh
./sleeper.sh 2
./sleeper.sh 1
./sleeper.sh 3
1
2
3
```

Tasks

1. Obtain the image set which will serve as an input for processing
    a. A set of 1000 images are available here:
    /net/pr2/projects/plgrid/plgglscclass/image_data_sets/data/training_images/
    b. please copy the images to a new directory in $SCRATCH
2. Create a converter.sh script which will convert an image using imagemagic:
    a. magick convert -adaptive-resize 3840x2160 -adaptive-sharpen 10 <input> <output>
    (processing of single image from t should take about 30s)
    b. Can be done in bash or any other scripting language
    c. The script must take “imagename” as an argument, the output can be determined based on inputfile or can be given as a second parameter
    d. It is best to test the conversion inside of an interactive job so login nodes is not stressed
        i. Lab02 shows and example how to start an interactive job in the plgrid-now partition.
3. Create a job script and execute it, the job should:
    a. Process in parallel multiple images inside of a single job
    b. Use the converter script from previous point
    c. Parallelization should be done using xargs ... srun combination
    d. Determine the proper –P parameter for xargs
    e. Determine the proper srun arguments
        i. Each process should be a single task using 1 core
    f. How to create a convenient input for xargs?
    g. Please add the:
    export OMP_NUM_THREADS=1
    right after the #SBATCH directives, this will instruct imagemagic to use single core per process.
    h. The job template is included below:

```
#!/bin/bash -l
#SBATCH --nodes=1
#SBATCH --ntask-per-node=48
#SBATCH --cpus-per-task=1
#SBATCH --account=plgglscclass24
#SBATCH --partition=plgrid-now
#SBATCH --time=01:00:00
# above config is mandatory!

export OMP_NUM_THREADS=1

# modules initialization
...

# data handling and work
...

# end of the script
```