Sequential executions

The minimum reserve unit to execute in TeideHPC is the node, so, regardless of the work to be executed, when accounting for it, it will be considered that full nodes have been used. That is, **the hours of use of the infrastructure is given by the number of hours that each node is executing the work in question, whether or not all the available resources are used.**

So that the user is not harmed by this fact, he will have to structure his executions so that he can group them in the same number of cores as the nodes (16).

Sequential executions usually require many job executions at a few cores, so the user must structure the input data for the executions in folders or file names identified by a job number, so that it can be easily manipulated.

Once the input data is organized in this way, it can be executed, launching as many tasks as fit in a node in a submit script.

Example of sequential execution of 1-core jobs

The following example script launches 16 1-core jobs simultaneously.

```
# !/bin/bash

#SBATCH -J <job_name>
#SBATCH -p <partition>
#SBATCH -N 1
#SBATCH --constrains=<node arquitecture>  # sandy, ilk (icelake)... arquitecture
#SBATCH -t <days-HH:MM:SS>
#SBATCH -o <file.out>
#SBATCH -D .
#############################################################

module purge
module load <modules>

for i in {1..16}; do
   srun --ntasks=1 --exclusive --output=slurm-%J.out serial-program --input /path/to/input.$i
&
done
# wait until all background processes are ended
wait
```

The `srun` command will launch individual jobs on different cores. The `&` symbol at the end of the line means that the job runs in the background, so that there is

no need to wait to launch the next job. The `wait` command at the end prevents the job from finishing until all previous processes have finished.

Example of sequential execution of 4-core jobs

The following example script launches 4 jobs on 4 cores simultaneously for a total of 16 cores used:

```
# !/bin/bash

#SBATCH -J <job_name>
#SBATCH -p <partition>
#SBATCH -N 1
#SBATCH --constrains=<node arquitecture>  # sandy, ilk (icelake)... arquitecture
#SBATCH -t <days-HH:MM:SS>
#SBATCH -o <file.out>
#SBATCH -D .
#############################################################


module purge
module load <modules>

srun --ntasks=4 --exclusive --output=slurm-%J.out serial-program --input /path/to/input_1 &
srun --ntasks=4 --exclusive --output=slurm-%J.out serial-program --input /path/to/input_2 &
srun --ntasks=4 --exclusive --output=slurm-%J.out serial-program --input /path/to/input_3 &
srun --ntasks=4 --exclusive --output=slurm-%J.out serial-program --input /path/to/input_4 &

# wait until all background processes are ended
wait
```