How to run jobs

Launch an execution

Sending a job to Slurm can be done in 3 different ways: using an interactive session, launching the application in real time or by means of an execution script.

In any case, you should always tell Slurm what resources you are requesting.

Execution of an interactive session

Using the `salloc` command, Slurm will give the user a compute node where, interactively, the user can work and run any software needed. This is a very useful option for testing new software or new data to work with, without the need to launch jobs to the queue, with the risk of them failing as soon as they start.

- Request a node:

```
salloc -N 1
```

- We request a node from a particular partition, assign a job name and duration to it:

```
salloc -p <partition> -J <job_name> -t <HH:MM:SS> -N 1
```

- Once the node and the job have been granted, you will receive the following message:

```
salloc: Granted job allocation 968978
```

- And when the job begins:

```
srun: Job 968978 step creation temporarily disabled, retrying
srun: Step created for job 968978
xxxxxx@node0101-1.hpc.iter.es's password:
Welcome to node0101-1. Deployment version 0.5-hpc. Deployed Wed May 13 19:58:32
WEST 2022.
```

Info

Once we have a node via `salloc`, we can, from another terminal, access to that node via ssh to have several sessions open on the same node and work on multiple things at the same time.

- To exit the interactive session:

```
[xxxxxx@node0101-1 ~]$ exit
```

Once we exit the node from the terminal where we did `salloc`, Slurm will release the node and the work will be mark as completed. Obviously, once we exit, all the extra sessions we have opened via SSH will be automatically closed.

Running a job in real time

With the `srun` command it is possible to launch a job to the queue directly.

```
srun -p <partition> -J <job_name> -t <days-HH:MM:SS> <aplicacion>
```

More available options:

Nombre	**Dirección IP**
-p <partition>	partition on which the jobs will run
-N <nodes>	number of nodes
-n=<num_tasks>	number of task
--tasks-per-node=<number>	task per node (consider -N)
-J <job_name>	job name
-t <days-HH:MM:SS>	expected time
-d=<type:job_id[:job_id]>	job dependency type and job dependency id (optional)
-o </path/to/file.out>	file for stdout(standard ouput stream)
-e </path/to/file.out>	file for sterr (standard error stream)

Nombre	Dirección IP
-D <directory>	default directory for execution
--mail-user=<email>	email for slurm notifications
--mail-type=<eventos>	event list notifications

Execute in SLURM via a script

The `sbatch` command sends a job to the queue to be executed by one or more nodes, depending on the resources that have been specified.

```
sbatch [-p <partition>] [-J <job_name>] [-t <days-HH:MM:SS>] mi_script.sh
```

The most basic structure for a script is as follows.

```
# !/bin/bash

#SBATCH -J <job_name>
#SBATCH -p <partition>
#SBATCH -N <nodes>
#SBATCH --tasks=<number>
#SBATCH --cpus-per-task=<number>
#SBATCH --constrains=<node arquitecture>  # sandy, ilk (icelake)... arquitecture
#SBATCH -t <days-HH:MM:SS>
#SBATCH -o <file.out>
#SBATCH -D .
#SBATCH --mail-user=<cuenta_de_correo>
#SBATCH –mail-type=BEGIN,END,FAIL,TIME_LIMIT_50,TIME_LIMIT_80,TIME_LIMIT_90
##############################################################


module purge
module load <modules>

srun <aplicacion>
```