Submitting a job

Once you have connected to the cluster, you can submit jobs to run on the cluster. A job has two parts:

- A script describing what the job does
- A description of the resources required to run the job - this describes things like the amount of memory and number of CPUs required by your job

The script should contain the commands required to run your job, and can be as simple or as complicated as you need. In the simplist case your script contains exactly the same commands that you would type on the command.

If you don't give a description of the resources, then the cluster will assume a default set of resources which is suitable for small jobs, but you should provide a more realistic estimate if you think your job requires more power. This is because the cluster assigns jobs to compute nodes based on these descriptions, and if you understate them, then the cluster may allocate more jobs that the node can handle and this will make your job run slower or even fail completely. Resource allocation is covered in more detail here.

Submitting jobs

After you first login to the cluster (as covered by the Connecting page) you are presented with a prompt something like the following:

```
[username@slurmsub ~]$
```

This is the command prompt. It shows you your username, the name of the computer you're logged into (submit) and the directory you are currently in (~ this symbol represents your home directory, which is the directory under which all of your personal files are kept). You can list the files in the directory by typing the command ls and then pressing the Enter key. If you do this you should see something like this:

```
[username@slurmsub ~]$ ls
example1.sh
[username@slurmsub ~]$
```

example1.sh is a file on the cluster which regards the old method of scripting. A script for the new method of submission would be:

```
#!/bin/bash
#Submit this script with: sbatch thefilename
#SBATCH --time=0:10:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -p batch   # partition(s)
#SBATCH --mem-per-cpu=100M   # memory per CPU core
#SBATCH -J "myjobname"   # job name
#SBATCH --mail-user=your@email.address   # email address
```

```
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --mail-type=FAIL
#SBATCH --qos=normal
# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
sleep 30
echo 'Hello World!'
```

Save this file as teste.slurm

Submit to the cluster using the sbatch command:

```
[username@slurmsub ~]$ sbatch teste.slurm
Submitted batch job 51293
[username@slurmsub ~]$ ll
-rw-r--r-- 1 username admin 545 Nov 11 13:11 teste.slurm
-rw-r--r-- 1 username admin   0 Nov 11 14:00 slurm-51293.out
```

The file “slurm-<job id>.out” gets the output of the script, in the example “slurm-51293.out”.

Job status

After a job is submitted, we can see its status by using the *“squeue -u <userid>”* command:

```
[username@slurmsub ~]$ squeue -u username
             JOBID PARTITION     NAME     USER    ST       TIME  NODES
NODELIST(REASON)
             51451     batch   myjobname username PD       0:00      1
(Priority)
```

Here we can see that the job is pending (represented by the 'PD' in the 'ST' column).

We can get information on the specific job with *“scontrol show job <job-id>”*

```
[username@slurmsub ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             51458     batch   myjobname username R       1:59      1
cfpsmall02
[username@slurmsub ~]$ scontrol show job 51458
JobId=51458 JobName=myjobname
   UserId=username(20314) GroupId=admin(1234) MCS_label=N/A
   Priority=2147406988 Nice=0 Account=(null) QOS=
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:01:46 TimeLimit=00:10:00 TimeMin=N/A
   SubmitTime=2019-11-11T14:28:08 EligibleTime=2019-11-11T14:28:08
   AccrueTime=2019-11-11T14:28:08
```

```
   StartTime=2019-11-11T14:28:22 EndTime=2019-11-11T14:38:22 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2019-11-11T14:28:22
   Partition=batch AllocNode:Sid=slurmsub.grid.fe.up.pt:2360
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=cfpsmall02
   BatchHost=cfpsmall02
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=100M,node=1,billing=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=100M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/homes/username/teste.slurm
   WorkDir=/homes/username
   StdErr=/homes/username/slurm-51458.out
   StdIn=/dev/null
   StdOut=/homes/username/slurm-51458.out
   Power=
```

The command *"s_squeue"* will show a brief summary of all the queues

```
[username@slurmsub ~]$ ./s_squeue
             JOBID PARTITION     NAME               USER    STATE SUBMIT_TIM
TIME TIME_LIMI  NODES   CPUS MIN_ FEATU NODELIST(REASON) PRI
             49958     batch   grblic           ge######  PENDING 2019-11-05
0:00      5:00      1      1 500M (null (Resources) 0.4
             51448     batch    reaKE        up2########  PENDING 2019-11-11
0:00 5-00:00:00      1      1   8G (null (Licenses) 0.4
             43213     batch script1.           de######  RUNNING 2019-10-11
31-00:29:52 33-08:00:00      1     16  16G (null ava08 0.9
             47901       big C250_600     ine###########  RUNNING 2019-10-28
13-17:38:36 20-00:00:00      1     20    0 (null inegi01 0.4
             51484     batch Structur    ine###########  RUNNING 2019-11-11
4:50  12:00:00      1     10   5G (null ava17 0.4
             ...
```

Anatomy of a script

nano teste.slurm

```
#!/bin/bash
#Submit this script with: sbatch thefilename
#SBATCH --time=0:10:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -p batch   # partition(s)
#SBATCH --mem-per-cpu=100M   # memory per CPU core
#SBATCH -J "myjobname"   # job name
#SBATCH --mail-user=your@email.address   # email address
```

```
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --mail-type=FAIL
#SBATCH --qos=test
# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
sleep 30
echo 'Hello World!'
```

This shows us several things. The first line *#!/bin/bash* is an instruction to the operating system to use the bash program to run the script.

The line sleep 30 is a command which simply does nothing for 30 seconds.

The final line echo “Hello World!” is responsible for creating the output we saw in *slurm-51293.out* - echo just prints out its arguments. The bash program is the same program that we have been running our commands in, so anything you type in your session can be put in a script and run. This is how we tell the cluster what to do - we put the commands we want to run in a script and submit it to the cluster, replacing sleep and echo with programs that do more useful work.

TIP: Use “*https://grid.fe.up.pt/web/index.php?page=slurm_job_script_generator*” for help generating the slurm script

Interactive Jobs

As well as submiting your jobs through scripts, it's possible to run a shell as a job using what's called an interactive job. This gives you a shell on one of the cluster nodes, exactly as you have on the submit host. This is ideal for doing interactive tasks such as compiling new software or doing data analysis. To invoke a interactive shell, simply use *“salloc”*:

```
[username@slurmsub ~]$ salloc
salloc: Pending job allocation 51496
salloc: job 51496 queued and waiting for resources
salloc: job 51496 has been allocated resources
salloc: Granted job allocation 51496
salloc: Waiting for resource configuration
salloc: Nodes cas04 are ready for job
```

To see Your jobs You can use *“sacct”*

```
[username@slurmsub ~]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
51293          myjobname    batch                     1  COMPLETED      0:0
51293.batch       batch                                1  COMPLETED      0:0
51293.extern     extern                                1  COMPLETED      0:0
51496              bash      batch                     1    RUNNING      0:0
51496.extern     extern                                1    RUNNING      0:0
```

To end the Interactive session just do *“exit”*

```
[username@slurmsub ~]$ exit
exit
salloc: Relinquishing job allocation 51496
```

Deleting and changing jobs

If you realize there is a problem after you've submitted a job, you can use the *"scancel <job-id>"* command to stop the job:

```
[username@slurmsub ~]$ sbatch teste.slurm
Submitted batch job 51498
[username@slurmsub ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             51498     batch  myjobname  username R       0:03      1 ava20
[username@slurmsub ~]$ scancel 51498
[username@slurmsub ~]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
51498          myjobname    batch                     1 CANCELLED+      0:0
51498.batch       batch                                1  CANCELLED     0:15
51498.extern     extern                                1  COMPLETED      0:0
```

Alternatively, if you just want to modify the job you can use the *"scontrol"* command. For details of the job properties that you can change, see the *"scontrol"* man page on the submit host (*"man scontrol"*)

```
scontrol update jobid=51458 ...
```