Cluster documentation - Hipatia

BCAM

October 2020

Índice

1. Introduction

Hipatia is the cluster name we have given it, but it is based in Slurm, some websites that may interest us with information about Slurm are:

- https://slurm.schedmd.com/overview.html
- https://support.ceci-hpc.be/doc/_contents/QuickStart/SubmittingJobs/SlurmTutorial.html

This cluster is developed under a Debian distribution and you need to have an account in order to login to the server, if in the following steps you get a login error, ask IT to create an account.

2. Connection

To connect to the server by using SSH on Linux and MAC, we've two options. By name:

Listing 1: Connection Hipatia

```
ssh -p 6556 <username>@hpc.bcamath.org
```

or directly by IP:

Listing 2: Connection Hipatia

```
ssh <username>@150.241.212.41 -p6556
```

After this, we're connected in our personal directory, "/home/<username>".

3. Internal structure - Hipatia

3.1. Personal directories - Hipatia

We've available mainly two personal directories to work in them:

- "/home/<username>/"⇒ We will usually work in this directory, so we'll transfer the files/folders from local to here and then run the Slurm commands to run our application
- "/workspace/scratch/users/<username>/"⇒ In case that we work with a folder structure with local references, it's advisable to work directly here.

3.2. Partitions Hipatia

Inside the cluster, we have the possibility of running our project in different partitions, to see what options we have, we execute the command "sinfo". Here we see the following partitions

- short ⇒ for jobs that may take up to 30 minutes at most
- medium ⇒ for jobs that may take up to 6 hours at most
- large ⇒ for jobs that may take up to 5 days at most
- xlarge ⇒ for jobs that may take up to 30 days at most
- extra ⇒ for jobs that may take up to 90 days at most

4. Basic commands

Here we can see several basic commands once we are connected:

- To see our pending jobs ⇒ squ
- To see information about jobs located in the Slurm scheduling queue of a specific user ⇒ squeue -u <username>
- To see why a job is in that state ⇒ scontrol -d show job <JOBID> | grep Reason
- To cancel the execution of a job ⇒ scancel <jobid>
- To cancel all jobs of a specific user ⇒ scancel -u <username>
- View information about Slurm nodes and partitions ⇒ sinfo
- To see what modules are available to load ⇒ module avail
- To submit script for a later execution ⇒ sbatch
  Here we can set options such as the number of tasks, the maximum execution time, the partition we want to use, the number of nodes we want to use for this job, etc
- To create job allocation and launch a job step, that is to run parallel jobs ⇒ srun
- As Slurm is finally Linux, let's not forget that we can use all the Bash commands like: ls, cat, rm, mkdir, etc

5. File ".sl"

In order to simplify the commands that we write in the cluster to run our scripts, we create a file ".sl".
We recommend started with the configuration commands, for example we can see here:

- –time ⇒ Set a limit on the total run time of the job allocation, once elapsed, it will stop
- –output ⇒ File name to save the output. The default file name is "slurm-%j.out", where the " %jïs replaced by the job ID.
- –error ⇒ File name to save the errors. The default file name is "slurm-%j.out", where the " %jïs replaced by the job ID.
- –ntasks ⇒ Request the maximum ntasks be invoked on each core.
- –ntasks-per-node ⇒ Request that ntasks be invoked on each node. If used with the –ntasks option, the –ntasks option will take precedence and the –ntasks-per-node will be treated as a maximum count of tasks per node.

- –cpus-per-task ⇒ Advise the Slurm controller how many processors will require per task, without this option, the default value is 1 per task.
- –mem-per-cpu ⇒ Minimum memory required per allocated CPU. Default units are megabytes, but we can specified different units using the suffix [K—M—G—T]
- –partition ⇒ Request a specific partition for the execution of our program. If not specified, the default behavior is to allow the slurm controller to select the default partition as designated by the system administrator.
- –job-name ⇒ This is the name that will appear when we look at the work in progress

We can see more options in https://slurm.schedmd.com/sbatch.html
A real example of this section is:

Listing 3: Connection Hipatia

```
#SBATCH --time=00:30:00 #Walltime
#SBATCH --output=slurm- %j.out
#SBATCH --error=slurm- %j_err.txt
#SBATCH --ntasks=1 # number of tasks
#SBATCH --ntasks-per-node=1 #number of tasks per node
#SBATCH --cpus-per-task=1 # 4 OpenMP Threads
#SBATCH --mem-per-cpu=1G # memory/core
#SBATCH --partition=medium
#SBATCH --job-name=Python_proj
```

After the initial configuration, we can see the modules required by the program/script we are going to run and load, remind you that we can see the different modules available with command "module avail".
Here we can see an example to load Python 3.7:

Listing 4: Load module Python 3.7

```
module load Python/3.7.4-GCCcore-8.3.0j
```

Here we can see an example to load Matlab:

Listing 5: Load module MATLAB

```
module load MATLABj
```

Finally, we will write the command to run the script/scripts we want. An example could be:

Listing 6: Run file Python

```
srun python helloWorld.py
```

If you're going to run a Matlab script, here an example:

Listing 7: Run file Matlab

```
srun matlab -nodisplay -r "HEM 60 twice, exit"
```

This is enough to generate our "sl"file, but there is a lot of flexibility and we can develop it as complex as we want.
Beyond all the available configuration commands, we can perform file manipulation, run several scripts in a sequential way... there are no limits.
PS: Try to be careful with the resources requested to be in solidarity with the colleagues.

6. Examples

6.1. Hello World - Python

We are going to develop a script to print "Hello World."and the current version of Python.

Listing 8: helloWorld.py

```
import sys

print("Hello Word")
print(sys.version)
```

So, we'll need develop as well the file ".sl"to configure the Slurm run in Hipatia. This time, we're going to seet a limit on the total run time (6 hours),we're going to use just 500MB of memory and the name in the cluster will be "Python_proj", we're going to save all the outputs and the errors that come out while the script is running, as our script is very simple, we don't need more than 1 cpu per task and finally we are going to run it on partition "medium".
We are going to load the module "python 3.7"with the command "module load...."and we finally write the command to run the python script with "srun python..."
We've the code:

Listing 9: testHipatia.sl

```
#!/bin/bash
#SBATCH --time=6:00:00 # Walltime
#SBATCH --mem-per-cpu=500M # memory/cpu
#SBATCH --job-name=Python_proj # CAREFUL TO CHANGE IT
ALSO IN THE RUN LINE
#SBATCH --output=slurm- %j_out.txt
#SBATCH --error=slurm- %j_err.txt
#SBATCH --cpus-per-task=1
#SBATCH --partition=medium

module load Python/3.7.4-GCCcore-8.3.0
srun python helloWorld.py
```

Now, all that remains is to transfer these two files to the cluster, for example with the command "scp", execute the script with the command:

Listing 10: scp example file

```
sbatch testHipatiaL.sl
```

We'll receive a message with the job ID like "Submitted batch job 1170397".
We can check the job with the command "squ":

Listing 11: scp command

```
             JOBID PARTITION PRIOR NAME USER STATE TIME TIME_LIMIT
NODES CPUS TRES_P START_TIME NODELIST(REASON) QOS
             1170397 medium 14801 Python_p adiaz RUNNING 0:01 6:00:00 1 1 N/A
2020-10-26T15:13:21 n001 normal
```

When our job is finished, we will see the errors file, "1170397_err.txt.ªnd the outputs file "1170397_out.txt", an easy way to read them is with the command çat namefile.txt".
In our example, we have said that it will generate the output and error files starting with the ID that Hipatia have assigned to the execution of the script, so that we can difference easily the different executions.
We can read the output file, for example with command çat", like çat 1159037_err.txt.ªnd çat 1159037_out.txt".

6.2. Example file ".sl' Matlab

An example of file ".sl"for Matlab:

Listing 12: scp command

```
             #!/bin/bash
             #SBATCH --time=6:00:00 # Walltime
             #SBATCH --mem-per-cpu=800M # memory/cpu
             #SBATCH --job-name=Matlab_proj
             #SBATCH --output= %j_out.txt
             #SBATCH --error= %j_err.txt
             #SBATCH --cpus-per-task=4
             #SBATCH --partition=medium

             module load MATLAB
             ## run Matlab
             srun matlab -nodisplay -r "matlab_parfor.m, exit"
             ## option -noFigureWindows allows to create and save figures without
opening figure
```

This will run the script "matlab_parfor.m".

7. Appendix 1: Copying working structure

To transfer files between local and server, here we suggest the SCP command that allows you to securely copy files and directories between two locations.
When transferring data with scp, both the files and password are encrypted so that anyone snooping on the traffic doesn't get anything sensitive.
If you're not familiar with the commands and you prefer use a visual software, you can use Filezilla to transfer all the files and directories between local-server and server-local.
We need hence, use the command as follow:

Listing 13: scp command

```
scp [OPTION] [user@]SRC_HOST:]file1 [user@]DEST_HOST:]file2
```

To transfer a file from local to server (remote) or from server to local, you will be asked for a user name and password.
An example of a single file:

Listing 14: scp example file

```
scp -P 6556 "/home/adiaz/Documents/MRC_bcam/minimax-hipatia.sl"
adiaz@hpc.bcamath.org:/home/adiaz/minimax-risk-classifier
```

The argument Pïs to specify the port.
An example of copy a directory recursively (just add argument r"):

Listing 15: scp example directory

```
scp -P 6556 -r "/home/adiaz/Documents/minimax-risk-classifier/"
adiaz@hpc.bcamath.org:/home/adiaz/
```

8. Appendix 2: Use CVX in Matlab

If you need to use CVX in Matlab, Go to http://cvxr.com/cvx/download/ and download CVX.
Put all your matlab files(main and functions) and the file HEM 60 twice.sl in the folder cvx, that will be automatically created once you download CVX from the website.
N.B.The main of the matlab file shoud start with the following command cvx setup
Transfer all the files Matlab and the file HEM 60 twice.sl (or the entire cvx folder with your matlab files and the file HEM 60 twice.sl) into the cluster, using FileZilla by dragging and dropping the files from your Local site (left in the FileZilla interface) to the Remote site (right in the FileZilla interface) in the folder named as your username.
Now that you have upload your files into the cluster, you are ready to run them.