HPC Tutorial: Basics I

Juan Pablo Mallarino*
Facultad de Ciencias – HPC, Universidad de los Andes - Bogotá, Colombia
(Dated: September 13, 2016)

I. THE HPC CLUSTER

The first step to use the services offered by the computing center is to establish a connection to the HPC cluster (i.e. *clustergate.uniandes.edu.co*). This is accomplished using the SSH (Secure SHell) protocol. If you are using a Unix based operating system (Linux, OS/X, or the like) you need to open the Terminal; e.g. in OS/X the terminal can be accessed under Utilities/Terminal from the application panel or by searching through spotlight (accessed pressing CMD+space-bar). If you are using any version of Windows there are several applications that provide an SSH connection protocol such as PuTTY. The following notes explain how to do this.

For information on the characteristics of the service and other important facts please refer to the User Policy.

Terminal Note I.1

Logging in from the terminal is quite simple. The terminal is a programming environment from where you can administer your computer and files, run tasks, and other cool stuff. It provides a console where you input commands using the syntaxis depending on the interpreter. Most common interpreters are *csh*, *tcsh*, *sh* and *bash*. Each of the previous interpreters provides different ways of issuing commands and accomplishing tasks. For simplicity, the default terminal in **almost** all terminal consoles is *bash* and it is the default environment running on the HPC cluster. Follow this link for information on most common commands, their usage and their advantages, which, in time, will become powerful tools for scripting, organizing/sorting directories/files, etc.

Connecting from the terminal (*do not type the $ sign into the console*)

```
$   ssh user@clustergate.uniandes.edu.co
```

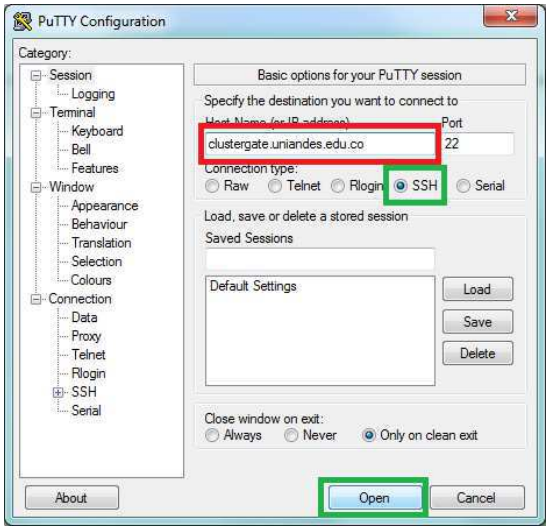
If you are within the University campus you can use **clustergate** instead of **clustergate.uniandes.edu.co**. Replace *user* with your user id and execute. If it is the first time you access the server it will request to validate the server certificate. After doing so, it will ask for your password. Once you are signed in, you will be forwarded to the cluster's terminal prompt.

PuTTY Note I.1

Logging in to the cluster using PuTTY or any other application from windows is straight forward. Figure 1 illustrates the procedure using PuTTY. It is recommended to read Terminal Note I.1 for additional information and helpful links.

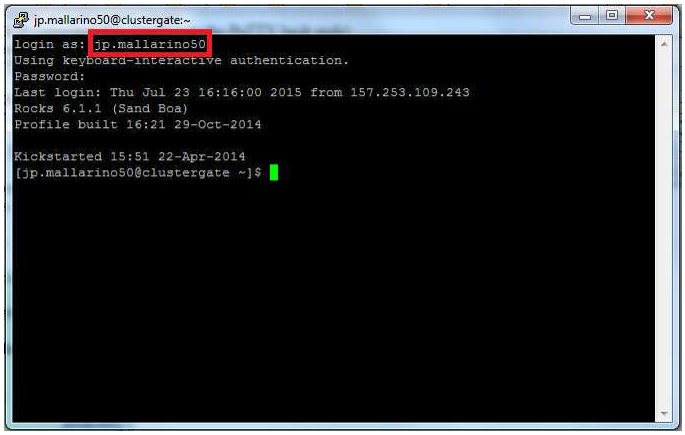
Once you have logged in to the server we are ready to submit any numerical computation but before digging into this topic please refer to the next section for tips & tricks as well as recommended practices.

* jp.mallarino50@uniandes.edu.co



(a) Server Address & Protocol

(b) Certificate



(c) User & Password

Figure 1: Procedure to log to the server using PuTTY.

II. TIPS, TRICKS & BEST PRACTICES

A. Autocomplete

One interesting feature of the terminal console is the ability to autocomplete triggered by pressing the **TAB** key from your keyboard. Autocompletion works as a filter for both commands and file/folder paths and it is recommended to use frequently as it helps to avoid typos. The following note illustrates how to use it.

Terminal Note II.1

For this simple example assume we are interested in the command *qnodes*, which shows the status of the computing nodes in the cluster, and you want to use autocompletion. In this scenario the first filter is the letter *q* that will reduce the possible commands to those beginning with that character. What will it show if you press the **TAB** key once?

Autocompletion: Part I (*do not type the $ sign into the console*)

```
$    q
```

The answer is nothing. The reason being is that the terminal recognizes either multiple commands

beginning with the letter *q* or none. When this happens, we can press the **TAB** key a second time, and the output will be as follows.

Autocompletion: Part II (*do not type the $ sign into the console*)

```
$    q
qalter               qmlplugindump qstop
qchkpt               qmlviewer     qsub
qcollectiongenerator qmove         qtconfig
qdbus                qmsg          qtconfig-qt4
qdbusviewer          qnoded        qterm
qdel                 qnodes        qttracereplay
qdisable             qorder        quota
qenable              qrencode      quotacheck
qgpumode             qrerun        quota_nld
qgpureset            qrls          quotaoff
qhelpconverter       qrun          quotaon
qhelpgenerator       qselect       quotastats
qhold                qsig          quotasync
qmake                qstart        qv_to_bqv.py
qmgr                 qstat
$    q
```

In order to refine the filter, type the *n* character and press **TAB** once again.

Autocompletion: Part III (*do not type the $ sign into the console*)

```
$    qnode
```

You may notice that in doing so it will autocomplete up *qnode* since there are two possible commands: *qnodes* and *qnoded*. If *qnoted* did not existed, autocomplete would have given *qnodes* and an extra space. Finally we require to type the *s* character and we are done. execute the command and see what happens!

This simple example demonstrates how to use autocompletion for commands. The same principles apply for file/folder paths. In addition to the versatility that this feature offers to the user, can you think of any other potential advantages?

B. Compressing/Extracting

The most common compression/packaging formats in Unix are **ZIP**, **GZIP**, **BZIP2**, and **TAR**. The former is quite useful for packaging files and folders. The remaining are compressing algorithms.

Terminal Note II.2

Compressing

Packaging log files, and folders *res1*, *res2* and *res3* using TAR to file *results_2015.tar* (*do not type the $ sign into the console*)

```
$    tar cvf output.tar files/folders
files/folders is a list separated by spaces.
$    tar cvf results_2015.tar *.log res1 res2 res3
```

Compressing *sim1D.log* using GZIP: output is *sim1D.log.gz* (*do not type the $ sign into the console*)

```
$    gzip sim1D.log
```

Compressing *sim1D.log* using BZIP2: output is *sim1D.log.bz2* (*do not type the $ sign into the console*)

```
$    bzip2 sim1D.log
```

You can also package and compress simultaneously using **TAR**.

Packaging and compressing log files, and folders *res1*, *res2* and *res3* using TAR and GZIP to file *results_2015.tar.gz* (*do not type the $ sign into the console*)

```
$    tar zcvf results_2015.tar.gz *.log res1 res2 res3
```

Packaging and compressing log files, and folders *res1*, *res2* and *res3* using TAR and BZIP2 to file *results_2015.tar.bz2* (*do not type the $ sign into the console*)

```
$    tar jcvf results_2015.tar.bz2 *.log res1 res2 res3
```

Terminal Note II.3

Extracting

Extracting the TAR file *results_2015.tar* (*do not type the $ sign into the console*)

```
$    tar xvf results_2015.tar
```

Extracting the GZIP file *sim1D.log.gz* (*do not type the $ sign into the console*)

```
$    gunzip sim1D.log.gz
```

Extracting the BZIP2 file *sim1D.log.bz2* (*do not type the $ sign into the console*)

```
$    gunzip sim1D.log.bz2
```

You can also extract packaged and compressed files simultaneously using **TAR**.

Extracting the TAR/GZIP file *results_2015.tar.gz* (*do not type the $ sign into the console*)

```
$    tar zxvf results_2015.tar.gz
```

Extracting the TAR/BZIP2 file *results_2015.tar.bz2* (*do not type the $ sign into the console*)

```
$    tar jxvf results_2015.tar.bz2
```

The question that remains is how to use **ZIP** format.

C. Optimize SSH

Optimizing SSH is quite important when you have bandwidth limitations or if you would like to manage with a different cipher protocol for security reasons. The standard protocol is *aes128-ctr* (safe) but there are several others alike, e.g. AES. We recommend *aes256-ctr* for increased security and the default for fast transfers. If bandwidth is a real issue and security is not the *blowfish-cdc* cipher could be used. An additional meassure for optimizing SSH connections is to use compression. The fallback of compression is the use of CPU on both the client and server side.

Optimized SSH for high security with compression (*do not type the $ sign into the console*)

```
$    ssh -c aes256-ctr -C user@clustergate.uniandes.edu.co
```

Optimized SSH for high security with medium security failover and compression (*do not type the $ sign into the console*)

```
$    ssh -c aes256-ctr,aes192-ctr -C user@clustergate.uniandes.edu.co
```

Optimized SSH for fast connections and little security with compression (*do not type the $ sign into the console*)

```
$    ssh -c blowfish-cdc,aes128-ctr -C user@clustergate.uniandes.edu.co
```

D. Transfering files

Copying files to and from the server is quite simple as it holds the same principles as the SSH connection. There are applications that have this ability such as FileZilla that has flavors for all known operating systems. However, most of the times it is more useful to learn the command line instruction.

Copy a file (*source.tar.gz*) from the local machine to the server into the home folder *app/* (*do not type the $ sign into the console*)

```
$    scp source.tar.gz user@clustergate.uniandes.edu.co:app/.
```

Copy the folder (*app_sources/*) recursively from the local machine to the server into the home folder *app/* (*do not type the $ sign into the console*)

```
$    scp -r app_sources user@clustergate.uniandes.edu.co:app/.
```

Copy a file (*source.tar.gz*) from the server to the local machine (*do not type the $ sign into the console*)

```
$    scp user@clustergate.uniandes.edu.co:source.tar.gz ./
```

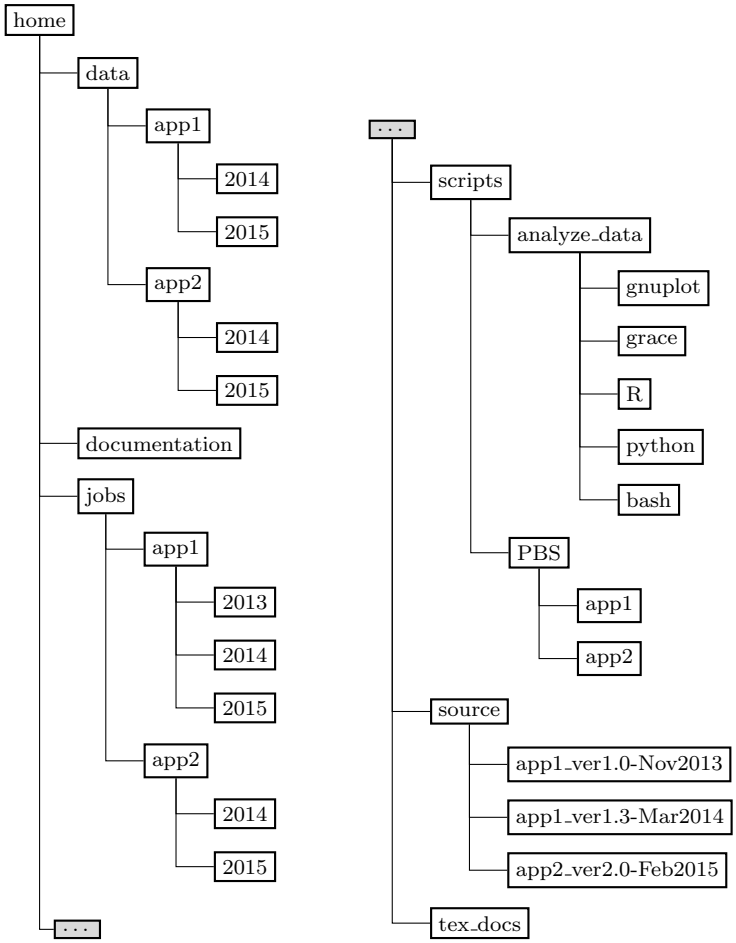
Copy the folder (*app_sources/*) recursively from the server to the local machine (*do not type the $ sign into the console*)

```
$    scp -r user@clustergate.uniandes.edu.co:app_sources ./
```

Note: A final remark on file/folder transfers is the ability to use the same optimizations as in the SSH connection with the same syntax; i.e. *-C* for compression and *-c cipher1,cipher2* for the cipher algorithm.

E. Best Practices

† Organize your data: although this may seem obvious, keeping the data sorted into an organized folder tree is time consuming and sometimes a challenge as the entropy tends to increase making data management and searching difficult tasks. A general rule of thumb is to create folder trees with topic/subject specifications categorized by year and purpose. For example,



† Avoid using spaces or special characters for file and folder names as Unix sistems and the file system are not fond of these. Instead of using spaces try using _, - and the period sign (.).

† Optimize disk-space by eliminating unnecessary files and compress text or log files. Also discern the relevant data from the bulk, as well as save all checkpoints for future computations.

‡ Avoid transferring large numbers of files to/from the server. Transfers are much faster and safer if the data is compressed initially and transfered afterwards.

□ Build scripts to automate tasks. This topic will be addressed in the future.

III. SUBMITTING A JOB

In order to use the HPC cluster we should submit jobs to the queuing system. The application will assign a computing node and resources (CPU, RAM and Walltime), as demanded by the user according to the User Policy queue schematics on Sec. 3.7, to perform the calculations. Our job distribution software is the well known *Torque/MOAB* used widely in HPC centers all around the world.

There are two ways that we can submit a job: i) interactive and ii) automated. Since the access to the cluster is restricted to the gate (*clustergate.uniandes.edu.co*) the scheduler sends the numerical execution information to the nodes via the job handling system. In automated job submission mode, the user prepares an execution script and the data to be transfered to the node for processing. In interactive mode, the user can start a live connection to one of the nodes and perform the computation directly. Interactive mode is usually recommended for short time executions and tests while long time and production executions are recommended with the automated modality. We illustrate both scenarios in the following notes.

The basic set of commands that the user requires to understand for the job handling system are as follows:

1. *qnodes* - Displays the full information of the nodes including processor type, total RAM, ongoing jobs and free resources.

2. *qstat* - Displays the status of either ongoing jobs or finished ones.

3. *qsub* - It is the basic command to submit an interactive or automated job

4. *qdel* - This command cancels a job. To use it you need to tell the scheduler the job-id to be terminated.

The *qsub* command admits a series of input arguments to specify the type and amount of resources that the user requires for the execution.

arg	**Description**
-q *queue_name*	(Depracated) This is the name of the queue to use. The setup we have implemented with the scheduler chooses this value automatically for the user according to the requested resources. In the past this line was necessary but not anymore.
-N *job_name*	The name of the job. This parameter is set by the user and it will be the name appearing in the job queue (command *qstat*). It is recommended to use keywords that reference to the type of execution, number of cycles, etc.
-M *email*	Specifies the email addres that the job handling system will use for notifications
-m *abe*	Makes the job handling system send (a)bort, (b)egin and (e)xit notifications to the specified email (-M)
-l walltime=*time*	Manually specify the computation resources: walltime, # of nodes or node (e.g. *node-11.local*), processors per node (ppn), and memory to be used in *mb* or *gb*
-l nodes=#:ppn=#	
-l mem=#mb	
-W	**Special** instruction that has several uses.
-I	The instruction to start interactive mode

Terminal Note III.1

To submit a job in interactive mode follow this example.

Interactive mode (*do not type the $ sign into the console*)

```
$    qsub -I -l nodes=node-3.local
qsub:  waiting for job 3122.clustermaster.uniandes.edu.co to start
qsub:  job 3122.clustermaster.uniandes.edu.co ready

[user@node-3 ~]$
```

Once in the interactive mode you can execute any computation. Can you think of other applications of the interactive mode?

Using the automated mode for job submission is the best course for lengthy or production value computations. In order to do this, we must build a script. The previous table shows all instructions that can be sent to the *qsub* routine. But the most powerful feature of the submit command is the ability to take a text file with the sequence of instructions embedded. We will review an example to ellaborate on this.

The HPC cluster contains a large number of software applications and libraries installed and will continue to extend the list with time. How can we access this list? There is a command line utility that provides this information; it is called *Modules*. Each application has its own requirements and in order to fulfill them without colliding with other software requisites it is necessary to build them within an encapsulated environment. Each of these environments can be accessed with the *Modules* command.

Terminal Note III.2

To list the available applications/libraries execute the following

Listing modules (*do not type the $ sign into the console*)

```
$    module avail
```

It will display in the console the complete list with the names of each of the applications. For example, if you need Gromacs you can verify that there are two available versions. Both of them work inside their environment. If you need v4.6.5 then we need to load the item *gromacs/4.6.5*. How do we do that?

Loading Gromacs (*do not type the $ sign into the console*)

```
$    module load gromacs/4.6.5
```

After you execute the above command all environment variables will be created in the background and you will be able to use it as if it were available globally. How do we unload a module?

Script Note III.1

Now that we know how to load modules we are ready to build our script to run a computation in automated mode. A script is a plain text file with a sequence of instructions that the terminal executes in an orderly fashion. It is composed by a heading and the body. The latter can be split into different sections according to the purpose of the script. In this case, the body will be split into five parts: i) definition of variables, ii) a useful optional debugging output (*script log*), iii) moving files to the node, iv) executing the computation, and v) moving files back from the node to the output folder. Tip: to avoid confusion between files use a keyword prefix or suffix for the script file; we suggest *script_name**.pbs***

Heading

```
#   This tells the terminal to use bash as the interpreter
#!/bin/bash
#   This is a comment. The heading of the script must include
#   instructions to the scheduler. The arguments begin with #PBS and
#   the options

#   First we need to stablish the name of the queue we will utilize.
#   For example if we need 8 intel processors and 64GB or RAM for one week in one node.
#PBS -l nodes=1:intel:ppn=8, mem=64gb
#PBS -l walltime=7:00:00:00
#PBS -M user@uniandes.edu.co
#PBS -m abe
#PBS -N test1_gromacs
#PBS -j oe
#PBS -o joboutput.txt
```

Body Part I

```
#   Create a temporary directory locally on the node optimizes I/O performance
TEMP_DIR=/state/partition1/$USER/$PBS_JOBNAME.$PBS_JOBID
#   Create the directory to store the computation output. For example
#   results inside the working directory.
OUT_DIR=$PBS_O_WORKDIR/results
```

Body Part II (Optional)

```
#   Creates a script log with the following information:
SCRIPT_LOG=script_$PBS_O_WORKDIR/$PBS_JOBNAME.$PBS_JOBID.log
touch $SCRIPT_LOG

#   Now store the information into the script log
echo "Job started on" `hostname` `date` >> $SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo "PBS working directory: $PBS_O_WORKDIR" >> $SCRIPT_LOG
echo "Current directory:" `pwd` >> $SCRIPT_LOG
echo "Temporary directory: "$TEMP_DIR >> $SCRIPT_LOG
echo "Output directory: "$OUT_DIR >> $SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo "User: $PBS_O_LOGNAME" >>$SCRIPT_LOG
echo "Batch job started on $PBS_O_HOST" >>$SCRIPT_LOG
echo "PBS job id: $PBS_JOBID" >>$SCRIPT_LOG
echo "PBS job name: $PBS_JOBNAME" >>$SCRIPT_LOG
echo "PBS environment: $PBS_ENVIRONMENT" >>$SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo >>$SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo "Full Environment:" >>$SCRIPT_LOG
printenv >>$SCRIPT_LOG
echo "#########################################" >> $SCRIPT_LOG
echo >>$SCRIPT_LOG
echo "The Job is being executed on the following node:" >>$SCRIPT_LOG
cat $PBS_NODEFILE >>$SCRIPT_LOG
```

Body Part III

```
mkdir -p $TEMP_DIR
mkdir -p $OUT_DIR
cp -Rf $PBS_O_WORKDIR/important_file(s) $TEMP_DIR/.
```

Body Part IV (Example: Running Gromacs v4.6.5)

```
#   We need to load the Gromacs v4.6.5 module
module load gromacs/4.6.5

#   execute gromacs using the configuration file config.txt
cd $TEMP_DIR
grompp configuration_file
```

Body Part V

```
cd $OUT_DIR
mv -f $TEMP_DIR ./
#   OPTIONAL (If you selected Body Part II): Output the finishing
#   date on the script log
echo "Job Finished: " `date` >>$SCRIPT_LOG
```

With this script Gromacs will be executed in any of the nodes designated automatically by the scheduler. To submit the job all you need to do is...

Submitting the Job (*do not type the $ sign into the console*)

```
$    qsub script_name
```

Important Note: Procure that all relevant configuration files and checkpoints reside in the same location as the script and submit the job from the same directory.

Although we have constructed a standard prototype of script to submit a job, there is an abundant documentation on the web with all kinds of interesting commands to improve and provide alternatives to all sorts of needs. Try searching the oracle!

1. http://qcd.phys.cmu.edu/QCDcluster/pbs/run_serial.html
2. https://www.msi.umn.edu/resources/job-submission-and-scheduling-pbs-scripts
3. https://hpcc.okstate.edu/content/software-resources
4. http://www.gla.ac.uk/services/it/hpcc/userguide/howtousetorquepbs/
5. http://www.physics.orst.edu/cluster_use

IV. ADVANCED TOPICS

How to use an application that is not installed in the application repository?