Computing Cluster User Guide

0. Basic rules

- The access to UOC Cluster, for the practical work of the course, will be realised by means of ssh (secure shell) connections, using a GNU/Linux distribution, or in other environments using adequate ssh client software, like for example Putty (a windows ssh client) and WinSCP (a scp/sftp client).
- The primary node of the cluster (frontend), where we connect by SSH, will **only** be used to make:
  - a) Compile code that will be launched later on compute nodes, by batch queues.
  - b) The final executions of tasks will be only realised into the batch queues of the system. In that sense, the users never execute tasks (in frontend or compute nodes) outside the queue system. **Running tasks outside the queuing system ARE NOT ALLOWED**, except for the tasks related to compilation of code in frontend node.
- The Tasks (Jobs) to execute, will be running over the queue system avalaible at the cluster, named OpenGridEngine (derived from SGE, Sun Grid Engine). The rest of this manual is oriented to provide some basic indications for the cluster users around how to execute tasks, and management of running tasks.
- All the running tasks in the queues, need to be of batch type, bucause this the applications need to be prepared for only use standard input and output, in the sense that if it is needed an input or output of data, is possible to realize that from redirecting the standard input or output from input/output files in the system.
- Finally, as a rules of «social etiquette», is mandatory among others:
  - In the system logout, observe at the frontend node, that didn't exist processes in execution or any standby or zombie status. Clear/stop all of the process users not related with the queue system.
  - If it is generated a temporally set of files, delete them at the end of their use period.
  - Store in the Cluster system, only files, data and applications related with the course activities.
  - In case of jobs (in queue) executing during long periods of time (e.g. various hours), is mandatory to logout the system, and explore periodically the execution state and their ending. In any case if they are errors of execution, or a they are executing a bit longer than the expected, is mandatory to remove the jobs from queues.
  - In that sense, sending to queues large jobs, it is recommended to test those jobs previously as a smaller jobs (putting them with les data, smaller matrix, smaller problem size, ...), for testing that the source code is correct, before execute more larger jobs.
  - If we observe errors or malfunctions in the system (hardware or software), please report to the email support address of the course.

In the next secction, we explain diferents aspects of the jobs execution and state management.

1. Submitting Jobs (qsub)

The queue system works under the OpenGridEngine (derived from the Sun Grid Engine, SGE).

We need to know, mainly the next commands: qsub , qstat , qdel; Which are briefly explained in the next sections of this guide. For more information (about options) see the manual pages for these commans (man pages).

The qsub command allows to send jobs to the batch queue system, qsub has the following syntax::
qsub [options] script [arguments]

Being *script* the pathname, either a binary file (in these case is needed the -b option) or (preferably) a script that contains the commanda that will run in the cluster nodes through a command shell. It is recommended this method for the pratical activities.

There are twho ways to send a job for execution

Method 1:
Add qsub options directly in the qsub command:
qsub -CWD -o output.log -i error.log job_script

Method 2: (**Recommended**)
The options of qsub are included in the script as a comments, that after interprets qsub and transmit to the shell, in that case the sending proces only needs:
qsub job_script

The contents of the job_script can be similar to the next example (the lines preceded by #$ are interpreted as options of the sending jobs to the queue system):

```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -o output.log
#$ -e error.log

. / your_commands
```

Description of the more common options of qsub:

Input / Output	
-o path_file	A standard output file, this file contains the output produced by the application. Within the script, they are some recognised variable names, that can help us to generate/name this file : $JOB_NAME, is the name of the job sended (the script name), and $JOB_ID is the identifier in the system of the created job. Is posible to create nomes like: -o out_$JOB_NAME.log$JOB_ID what left us the file with customized name for each job sended.
-e path_file	Standard error output filename, contains the occurred errors. Same variables (than -o option) can be used for creating the name.
-j yes | no	Add output and error output in the same file (the two last files in one

	mixed).
-i path	Standard input file (if is it needed), containing input for the task to be executed.
-cwd	Execute the task whitin the current directory, in fact all the previous files (I/O) are related to current work directory (cwd)
Other options	
-N name	Optional name of the task
-w v	Test if the syntax of the job is correct (didn't send the job)
Parallel Jobs / Parallel Environments	
-pe enviroment slots	Has to be specified, the parallel environment, and the number of resources needed (threads in a OpenMP case) for the application. For obtaining the avalaible parallel environments, use the command: **qconf -spl** In our case, one of them is: openmp — To be used when working with multithreaded applications (OpenMP applications).

In the next examples, we show the different procedures, depending on the task type, sequential or parallel.

1.1 Doing sequential tasks

qsub job_script
Contents of *job_script* can be similar to this example fragment:

```
#!/bin/bash

#Use Bash for the execution shell
#$ -S /bin/bash

# SGE uses current directory as a work directory for the used files.
# Output files (output.dat and error.dat) are produced in local directory
# Binary to execute is also found in the same directory.
#$ -cwd

# Redirecting standard output to this file.
#$ -o output.dat

# Redirecting standard error output to this file.
#$ -e error.dat

# This script (or binary command depending on the case) will be executed in the current
directory
# Ensure that the script/command has execute permission (chmod +x script.sh).
./script.sh
# or ./command if it is a binary program
```

1.2 Parallel OpenMP Jobs

qsub job_script
Contents of the *job_script* can be similar to:

```
#!/bin/bash

#Use Bash for the execution shell
#$ -S /bin/bash

# SGE uses current directory as a work directory for the used files.
# Output files (output.dat and error.dat) are produced in local directory
# Binary to execute is also found in the same directory.
#$ -cwd

# Redirecting standard output to this file.
#$ -o output.dat

# Redirecting standard error output to this file.
#$ -e error.dat

# Uses the parallel enviroment "openmp". The number used as parameter (4) is the numer of
cores reserved to the execution, initially could bu adjusted to (1) or (2) for using less cores for
testing purposes.
#$ -pe openmp 4

#It is needed to point that in OpenMP is avalaible de OMP_NUM_THREADS variable where
#we can indicate the number of threads avalaible to the aplication to execute.
#The variable $NSLOTS indicates the previous number (4 in this example) that we passed to
#openmp environment, in this case we have 1 core for each OpenMP thread.
#But is posible to increment the variable for having more threads per core, for example,
#if we have 4 cores we can put OMP_NUM_THREADS=8 , having with that 2threads/core

export OMP_NUM_THREADS=$NSLOTS
./openmp_executable
```

NOTE: Taking into account the max cores per compute node (4 cores in our cluster system), doesn't have any sense that we demand more cores than the avalaible per node (in the option -pe in the previous example). If this occurs, more cores demanded than available per node, simply the task didn't run, and it can be putted in the queue system indefinitely. In that case is better/needed to delete the task from queue system.

2. Job monitoring (qstat)

For obtaining information about the execution or jobs waiting, we need to use:
qstat [options]

For limiting the output of qstat we can execute **qstat | grep - v hqw** for filter only the pending jobs, or **qstat - sr** showing only the jobs in execution.

Other options of qstat :

-u user	Shows jobs submitted or in execution for the mentioned user.
-j id-job	Show information of a job identified by the number id-job.
-f	Shows all the queues and jobs
-help	Help about qstat summarising the options.

In the case of pending jobs, we can obtain some clues about the reason of the wait state, by executing
qalter -w p id-job

It is posible also verify a job with:
qalter -w v id-job

If that command show the beneath message, that means that the job have some problems in the queue system, and never can be completed the execution of the job, in that case we need to delete that task:

```
verification: no suitable queues
```

3. Deleting a Job (qdel)

For deleting a job, in case of error, malformed job, or some job that have more than expected time to execute, we can do:
qdel id-job