IQTC users guide

11/01/2021

This guide is addressed to IQTC users. It gives the basics for the correct usage of the calculation resources for the following clusters:

- **iqtc01**: AMD64 nodes - HP Proliant DL145 G2 (See nodes list)
- **iqtc02/03**: Intel EM64T nodes - HP Proliant DL160 G5 and HP Proliant DL140 G3 (See nodes list)
- **iqtc04**: Intel EM64T nodes - HP Proliant DL160 G6 (See nodes list)
- **iqtc05**: AMD Opteron 6276 – SGI H2106-G7 (See nodes list)
- **iqtc06**: AMD - HP ProLiant DL560 Gen 8, Supermicro SuperServer 8017R-TF (See node list)
- **iqtc07**: Supermicro server (See node list)
- **iqtc08:** HP Proliant DL360 Gen9 (See node list)
- **cerqt2**: Intel IA32 nodes - Sun Fire V60X, AMD64 nodes - Sun Fire V2 (See node list)

All of them are placed in the main datacenter

Users access to IQTC computational resources through a redundant access portal. This is a computing resource that allows users to be identified into the system and let them use the cluster file system, submit jobs and manage those which are already running.

From this portal users are able to access all of IQTC computing nodes. It is important to mention that each cluster has its own applications (at /aplic directory) and libraries (at /opt directory) compiled and optimized for its architecture. Moreover, each node has its own local disk (at /scratch directory), from where most of calculations must be done.

Access

1. Access to IQTC clusters

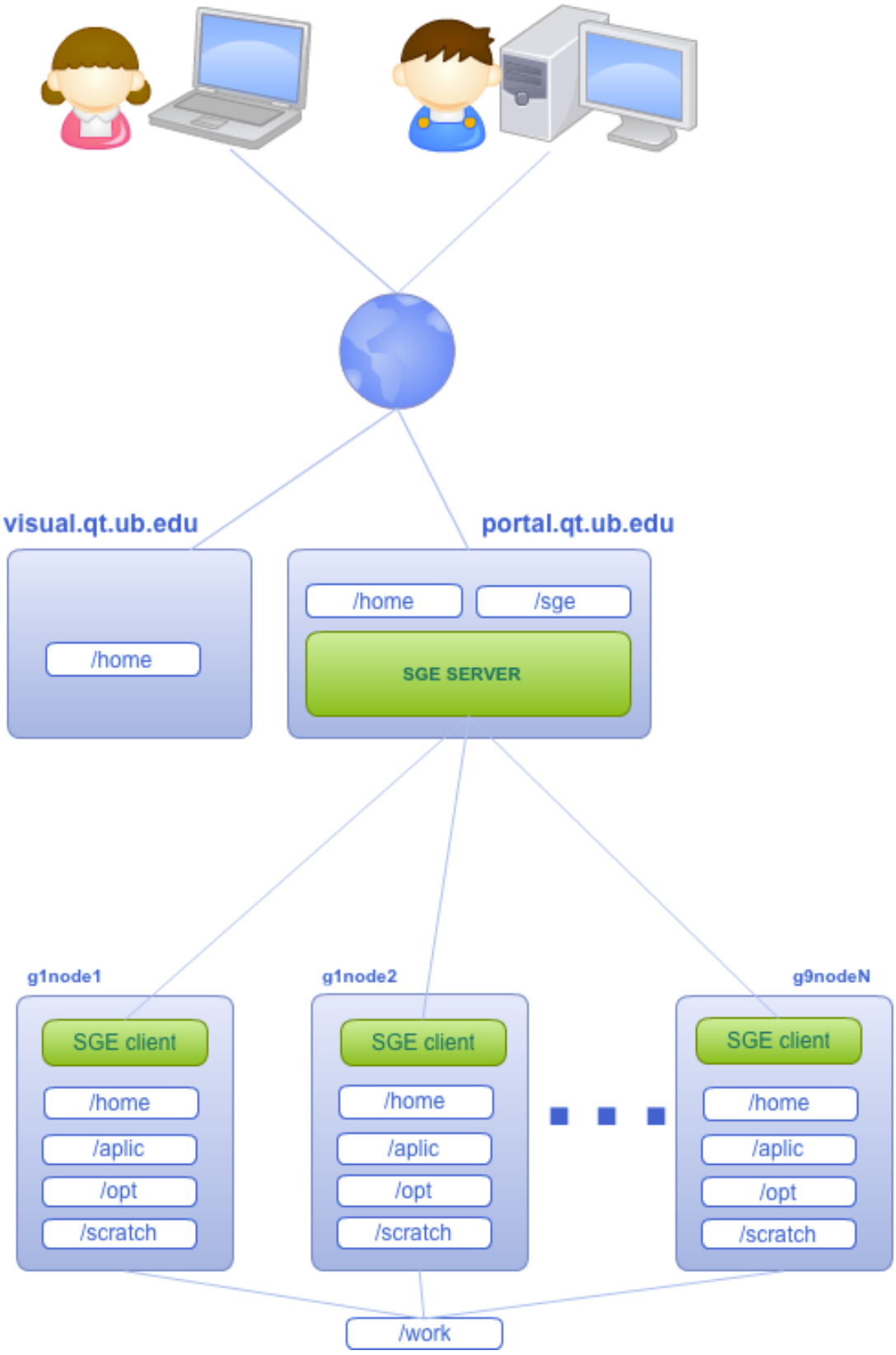
To login to the computational resources, a ssh session must be used to ***portal.qt.ub.edu*** server using the password provided by system administrators of IQTCUB. As you can see on the next figure, there is another portal where you can login into the system: ***visual.qt.ub.edu***. This is the visualization portal and it is reserved for the use of visual applications only. Notice that from this machine it is not possible to submit jobs to the clusters.

There is also another access portal that is reserved to transfer files between the cluster and your computer in order to not affect the rest of users, ***portalftp.qt.ub.edu***.

- **Portal** -> **portal.qt.ub.edu** -> To submit calculating jobs to the clusters
- **Visual** -> **visual.qt.ub.edu** -> To run visual applications. (click here for more info)
- **PortalFtp** -> **portalftp.qt.ub.edu** -> To transfer data between the cluster and your computer. (For more information see section: IQTC users guide: File System)

All of them acces to the same filesystem. This mean you will see the same files you have independently the access portal used.

Independently of the portal used, if the user's session stays inactive for 4 hours, the system will automatically close the ssh session to keep the security and performance level.



Portal is connected to the cluster file system (/home), which is exported to all clusters nodes and where users have their data. Therefore, this directory is available from all computing resources.

Each user has his own directory (/home/user) which is daily copied to the backup system, so is important to store all important data there. This directory is not a storage space, it is meant to be for the current calculation data, for this reason it is important that each user stores the results to their own local storage devices once the calculations are done.

Users can submit jobs to the SGE queue system from portal, and are able to check out which application are installed in each cluster using Environment Modules.

2. Access to CERQT2 cluster

TFG and Master users have access to a complete isolated clusters called CERQT2 (See hardware details here: The queues & clusters). Identically to IQTC clusters, there is an access portal that must be used in order to log to the system and be able to send jobs, this is ***cerqt2.qt.ub.edu***. There is also another access portal if you are planning to run graphical applications, ***practiques.qt.ub.edu*** .

- **Cerqt2** -> **cerqt2.qt.ub.edu** -> For TFG and Master users only
- **Practiques** -> **practiques.qt.ub.edu** -> To run graphical applications

Both of them acces to the same filesystem. This mean you will see the same files you have independently the access portal used.

Login

To login to IQTCUB's computational environment, OpenSSH protocol must be used with the following parameters:

HOST : portal.qt.ub.edu
Protocol : SSH2
PORT : 22

Depending on the operating system used, several applications are available to allow the OpenSSH connection:

Login from Linux/Unix

Login from Windows

Login from Windows for running remote graphic applications

1. Login from Linux/Unix


To log in, from Linux/Unix console, the applications *gnome-terminal, yakuake, xterm* or *tilda* are recommended because of their features. The following command shows how to login:

```
ssh -X user@portal.qt.ub.edu
```

where:

ssh*: protocol used*

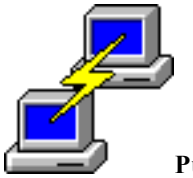
-X*: is a flag to allow visualization (notice that Mac OS systems uses the flag -Y instead of -X)*

user*: username provided by IQTC Sys Admins*

portal.qt.ub.edu*: the access portal. This may be replaced by visual.qt.ub.edu if only visual applications are going to be used.*

2. Login from Windows

There are several OpenSSH clients to use in Windows, for example:


Putty : http://www.chiark.greenend.org.uk/~sgtatham/putty/


Cygwin : http://cygwin.com/

Cywgin installation guide for SSH connection:
https://docs.oracle.com/cd/E24628_01/install.121/e22624/preinstall_req_cygwin_ssh.htm#EMBSC281

The use of these applications is similar to linux terminals.

3. Login from Windows for running remote graphic applications

If you are running a Windows operation system and need to execute a remote graphic application you will need to use Xming + Putty applications in your local machine. If you are interested in doing so, please review this user guide use remote X aplications with xming + putty

File System

1. File System Structure

Once logged to the system, users are able to access to the file system. Notice there are 6 directories exported that are accessible from each component of the IQTC system.

- **/home**

User data file system (/home/user) the only one with daily backup policy. Available from portals, visualization portal and computing nodes. The home for each user may be consulted through the environment variable **$HOME**. Notice, there is a quota per user of **150GB**. If a user needs more space temporally, he/she should contact with IQTC sys admins.

- **/work**

Temporal cluster calculation shared file system (/work/user), designed for files that must be shared by different nodes when the jobs are running in more than one node. Data older than 45 days will be deleted. Available from portals and computing nodes. The /work directory for each user may be consulted through the environment variable **$WORK**.

- **/scratch**

The computing nodes' local file system (/scratch/user). Each node has its own scratch and is faster to read/write than the others file systems. Used to calculate within just one computing node. When the job finishes the data must be copied to the /home/user directory and deleted. The scratch directory for each user may be consulted through the environment variable **$SCRATCH**.

- **/aplic**

Application file system. Each cluster has its own /aplic. Available from computing nodes.

- **/opt**

Libraries and compilers file system. Each cluster has its own /opt. Available from computing nodes.

- **/sge**

SGE file system. Contains Sun Grid Engine configuration files and binaries. This directory is referenced through the environment variable **$SGE_ROOT**. Available from portal and computing nodes.

There is a backup done every night of the home directory. For more information related to backups, please visit the storage system section. Please notice that there is no backup of /work nor /scratch directories.

2. Remote File System Access

In order to access to the data of your /home in the IQTCUB/CERQT2 clusters, we recomend you to use one of the following applications:

- For Linux:
  - **Terminal**: using SSH protocol. Type ssh user@host
  - **Filezilla**
  - **Konqueror** (KDE): type fish://user@host
  - **Nautilus** (Gnome): type sftp://user@host

- For Windows:
  - **WinSCP** http://winscp.net
  - **Filezilla** **https://filezilla-project.org/** (Find a filezilla user guide here)
  - **SSHFS** http://fuse.sourceforge.net/sshfs.htm

- For Mac:
  - **SSHFS** **http://code.google.com/p/macfuse/wiki/MACFUSE_FS_SSHFS**

Remember that if you are planning to transfer data between your computer and the remote file system, you should use as ***'host'*** *portalftp.qt.ub.edu* for data in IQTCUB clusters. For data in CERQT2 cluster, you may use the access portal, *cerqt2.qt.ub.edu*.

3. Remote File System Transfer

RSYNC over SSH protocol is the best choice to transfer big data volumes. See the command below:

- **Transfer data from your computer to the server:**

rsync -avHl --progress /path/origin/* sshserver:/path/destination/

IQTC example:

```
rsync -avHL --progress /home/username/carpetToCopy
username@portalftp.qt.ub.edu:/home/username/
```

CERQT2 exemple:

```
rsync -avHL --progress /home/username/carpetToCopy
username@cerqt2.qt.ub.edu:/home/username/
```

- Transfer data from the server to your computer:

rsync -avHl --progress sshserver:/path/destination//path/origin/

IQTC example:

```
rsync -avHL --progress
username@portalftp.qt.ub.edu:/home/username/carpetToCopy /home/usernam
e/
```

CERQT2 example:

```
rsync -avHL --progress
username@cerqt2.qt.ub.edu:/home/username/carpetToCopy /home/username/
```

Parallel enviroments (PE)

Parallel environment is a concept used to distinguish wether a job uses parallelization (distributing the data across different parallel computing nodes) or not.

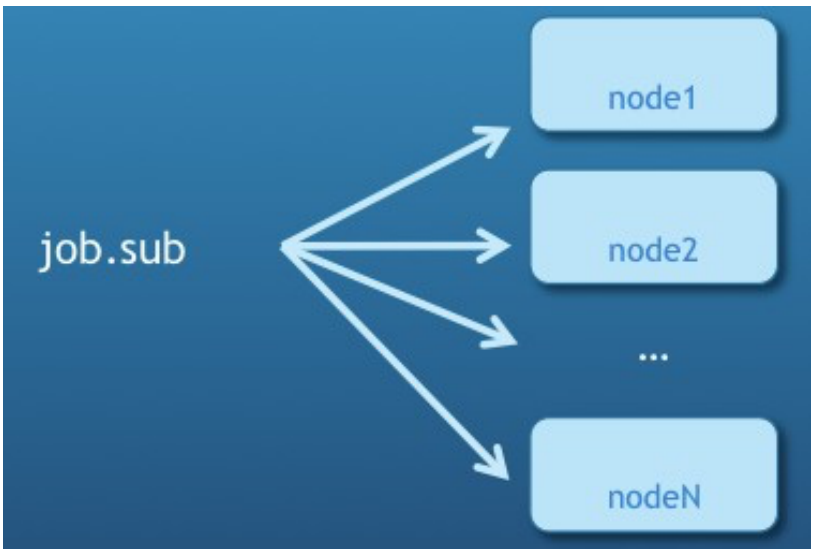
In IQTC there are several parallel enviroments available:

- **smp**

Parallel environment used for jobs that will be executed within one node. It may be able to parallelize but only using the cores inside the execution node.

- **omp***

Parallel environment used for parallel executions between different nodes using OpenMPI. This is the best option for more than one node parallel execution in IQTC clusters.



- **linda**

Parallel environment used for parallel executions between different nodes for Gaussian application only.

- **qchem2**

Used only for Qchem application for parallel executions between different nodes.

- **adf**

Used only for Adf application for parallel executions between different nodes.

An example of how to send a job with a particular parallel environment:

qsub -q iqtc04.q -pe **omp*** 24 job_name.sub

Modules

Module is a free software very useful to manage environment variables of each application and it is very easy to use. Each module, loads all environment needed by a certain application and all dependencies automatically. See more information here.

This is really useful when there are several versions of the same application. Loading the module associated to the application version planned to use, avoids conflicts with other versions when executing the application.

1. Available modules

1.1 Module list

There is a module for each application version available in a cluster. To see the complete list, please check applications section.

1.2 Available modules in Visual

In Visual server you can check the available modules using directly the command:

module av

1.3 Available modules in Portal

There are specific commands to obtain information about each cluster available modules. Replace X for the cluster number to be checked:

module_iqtc0X avail
lists available modules in iqtc0X cluster

module_iqtc0X show module_name - displays full information about the module with name "module_name" from iqtc0X cluster.

module_iqtc0X whatis module_name - displays some information about the module with name "module_name" from iqtc0X cluster.

In **CERQT2** cluster the commands are:

module_cerqt2 avail
module_cerqt2 show module_name
module_cerqt2 whatis module_name

Remember that access portal is not meant to be an execution host, then it is not possible to load modules in the portal.

Example of use:

```
user@portal02:~$ module_iqtc04 avail
------------------------------------------------------------------------
            Modules that we can find in the iqtc04 cluster
------------------------------------------------------------------------
------------------------------------------------------------------------
                              Compilers
------------------------------------------------------------------------
cmake/2.8.0  intel/9.1               intel_compiler_suite/11.1.059  pgi/10.2
intel/10.1   intel_compiler_suite/11.0.074  intel_compiler_suite/11.1.072  pgi/8.
0-6
------------------------------------------------------------------------
                              Libs
------------------------------------------------------------------------
alps/1.3.5_intel-11.1.072       hdf5/1.8.6_mpi      parallel_python/1.6.1
alps/2.0.0_intel-11.1.072       intel_mkl/11.0.074  petsc/3.0.0-p12_intel10.1
alps/2.0.0_p060312_intel-11.1.072  intel_mkl/11.1.059  petsc/3.0.0-
p12_intel10.1_complex
alps/2.0.0_p130312_intel-11.1.072  intel_mkl/11.1.072  petsc/3.1-p4_ics-11.1.072
fftw3/3.2.2               matplotlib/1.1.0    petsc/3.1-p4_ics-11.1.072_fftw3
fftw3/3.2.2_gcc-4.3.2           mkl/10.0           python-ase/3.5.1.2175
fftw3/3.2.2_ics-11.1.072       mkl/10.1           scipy/0.10.0
fftw3/3.2.2_pgi-10.2           mkl/9.1           slepc/3.0.0-p12_intel10.1
gsl/1.14_intel10.1          numpy/1.6.1
------------------------------------------------------------------------
                              MPI
------------------------------------------------------------------------
ofed/1.5.1                     openmpi/1.4.2_pgi-10.2_ofed-1.5.1_blcr-8.2
openmpi/1.4.2_ics-11.1.072_ofed-1.5.1_blcr-8.2    openmpi/1.4.3_gcc-4.3.2_ofed-
1.5.1
openmpi/1.4.2_intel-10.1_ofed-1.5.1_blcr-8.2
------------------------------------------------------------------------
                              Applications
------------------------------------------------------------------------
adf/2010.02                   jumbomem/2.1
adf/2012.01                   molcas/7.2
adf/2012.01a                   molcas/7.4.p45
amber/10_ompi                   molcas/7.4.p45_ompi
amber/11_ompi                   molcas/7.6.p51
blcr/0.8.2                  molpro/2009.1p20
charmm/c35b1r1_heap_size_ompi          msindo/3.5
charmm/c35b1r1_heap_size_ompi_serial  numactl/2.0.6
charmm/c35b1r1_ompi              nwchem/2011_oct_ompi
cpmd/3.15.3_ompi               nwchem/5.1.1_ompi
crystal/09_ompi                  nwchem/6.0_ompi
dftb+/081217                   nwchem/6.1_ompi
dl_poly/2.17_modfhl              octopus/4.0
dl_poly/2.17_ompi               openbabel/2.2.3_gcc
dl_poly/4.02_ompi               openmx/3.5_ompi
espresso/4.3.2_ompi               orca/2.7.0b_ompi-1.4.2
```

```
gamess/2006_ompi              orca/2.8.0.2
gamess/2010r2                  orca/2.9_ompi
gamess/2010r2_ompi              qchem/4.0
gamess-uk/7.0_ompi              qchem/4.0.0.1
gamess-uk/7.0_qmmm_ompi              qchem/4.0.0.1_mpich
gamess-uk/8.0.200312_ompi          qchem/4.0_mpich
gaussian/g03d02                  quantumwise/atk-10.8.2
gaussian/g09b01                  quantumwise/atk-11.8
gpaw/0.8.0.8092_ompi              quantumwise/atk-11.8_DFT
gromacs/3.3.1_ompi               quantumwise/atk-12.2
gromacs/4.0.2_localpressure_ompi      siesta/3.1_ompi
gromacs/4.0.7_fftw3_ompi          siesta/3.1.pl9_ompi
gromacs/4.0.7_mkl_ompi              siesta/ldau_ompi
gromacs/4.5.5_fftw3_ompi          siesta/trunk-367_ompi
gromacs/4.5.5_mkl_ompi              turbomole/5.10
inelastica/1.1                   turbomole/5.10_mpi
jaguar/7.7                   vasp/4.6.36_ompi
jaguar
```

In the above figure, you may see the output of using command *module_iqtc04 avail* from access portal. Notice there are **several sections: Compilers, Libs, MPI and Applications**. Each section contains all the versions available in the specified cluster.

1.4 Available modules in a node

module av - lists available modules

module show module_name - displays full information about the module with name "module_name".

Example of use:

```
 tarenal@g9noden4:~$ module show vasp/5.2.12_11nov11_ompi
-------------------------------------------------------------------
/opt/modules/modulefiles2/Applications/vasp/5.2.12_11nov11_ompi:
module-whatis  Carrega les variable d.entorn de VASP 5.2.12
prepend-path  PATH /aplic/vasp/5.2.12_11nov11/ics-11.1.072/ompi-1.4.2
module  load openmpi/1.4.2_ics-11.1.072_ofed-1.5.1_blcr-8.2
module  load intel_mkl/11.1.072
-------------------------------------------------------------------
```

As shown in the above figure, this command shows where is the module installed, a description, other variables needs to be setted and which other modules needs to be loaded.

module whatis module_name - displays some information about the module with name "module_name".

2. Load/unload module

This command is only available from a cluster node.

module load module_name - loads the module with name "module_name" and its dependencies. Example of use: module load vasp/5.2.12_11nov11_ompi

module unload module_name - unload the module with name "module_name" and its dependencies.

3. List loaded modules

This command is only available from a cluster node.

module list - list all modules currently loaded.

Example of use:

```
user@g9noden4:~$ module load
vasp/5.2.12_11nov11_ompi
user@g9noden4:~$ module list
Currently Loaded Modulefiles:
  1) intel_compiler_suite/11.1.072
  2) blcr/0.8.2
  3) ofed/1.5.1
  4) openmpi/1.4.2_ics-11.1.072_ofed-
1.5.1_blcr-8.2
  5) intel_mkl/11.1.072
  6) vasp/5.2.12_11nov11_ompi
```

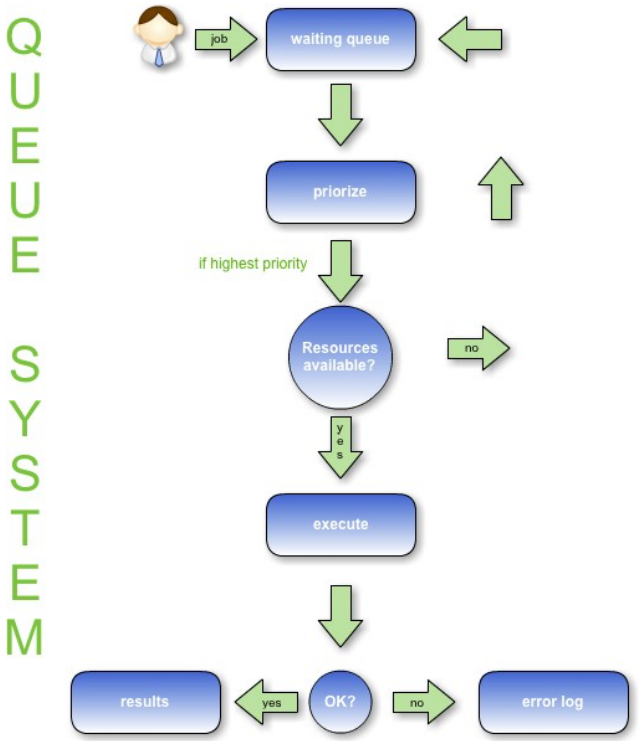
Queue system SGE

IQTC provides a queue system to manage calculating jobs within clusters. Within the different queue systems available, the one selected in IQTC is Sun Grid Engine (SGE).

This software manages the jobs sent by users in order to assign them a priority and find for them the best resources available. It also keeps jobs in a waiting queue if no resources are available until there are.

1. How it works

The procedure of a queue system is explained in the next figure:



1. User send a job via a script file and specific commands (see following chapters).
2. The queue system interprets the job and put it by default in a waiting queue.
3. The job gets a priority (see fair sharing section).
4. When the job becomes the one with the highest priority, the queue system looks for available resources in the specified computing nodes queue.
5. If there are enough resources available, the job is executed. If there aren't, the job is kept in waiting queue and the resources already available become reserved. Once there are again available resources, the job will be executed.
6. After the execution, results are saved and errors logged as specified in the script file

2. The queues

In IQTC, nodes are arranged into several queues depending on its hardware characteristics (memory, disk, etc.) By default, the queue gets a similar name as the cluster and finnishes with .q:

- **iqtc01.q**
- **iqtc02.q**
- **iqtc04.q**
- **iqtc05.q**
- **iqtc06.q**
- **iqtc07.q**
- **iqtc08.q**
- **cerqt2.q**

A complete list of all available queues and their hardware details may be found here.
There are two types of queue that determine an execution mode. These are the available execution modes:

2.1 Batch mode

This is probably the most habitual queue use. Users submit their jobs from the access portal to a certain queue with or without a resource requested and SGE runs it when the appropriate resources are available. This mode allows to send lots of jobs and lets the queue system to manage them.

The batch mode queues are named with its cluster name. For example: iqtc04.q

For more details on how to calculate in batch mode, see section SGE most used commands

2.2 Interactive mode

In this mode, users may login directly to a node with a special command and run there whatever they want. This mode is thought to be useful to compile, testing process, debug, profiling, etc. It is also available to calculate interactively but always having in mind that there is a time limit of 12 hours. After this time, the user will be logged off.

The interactive mode queues are name as follow: interactive0X.q where X corresponds to the number that identifies the cluster. These are the available interactive queues for clusters iqtc01, iqtc02 and iqtc04:

- ***interactive01.q***
- ***interactive02.q***
- ***interactive04.q***

NOTE: In case of iqtc06 cluster there isn't an interactive06.q queue, in this case you must use the same ***iqtc06.q*** *queue for using a node in interactive mode. There are no other interactive queues in the system.*

For more details on how to calculate in interactive mode, see section SGE most used commands

Fair sharing policy

1. Job priority and available resources

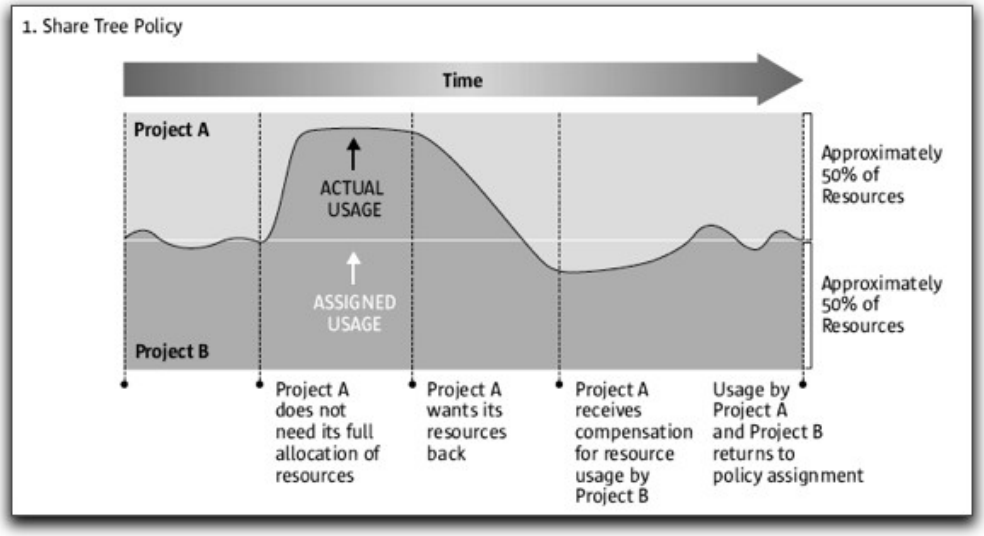
In order to decide which jobs will execute first and so, schedule them in queues, Sun Grid Engine (SGE) has several policies that may be applied. The one decided to be used in IQTC clusters is: Fair sharing policy.

This policy is NOT applied at CERQT2 cluster.

The Fair Sharing policy calculates a dynamic **job priority** for a group Y based on the accumulated usage during a configurable period of time and the amount of computational resources that each IQTC group has (*share*). This is calculated automatically by the SGE system.

$$priority_y = f(time, share_y)$$

Another characteristic of the Fair Sharing policy is that it offers the possibility of using more resources than they really own when they need it. This is done because SGE shares the computing resources of each group, when they are not used, to the rest of the groups. Therefore, users may be, during a certain time, calculating using more cores than those they really own. This is done calculating a share for each group in function of the computing nodes they own.



As seen in the above figure, at the moment that a group (called project A in the image) does not use their computing nodes, another group (B) with a major and punctual necessity, may use those free computing nodes. When the first group (A) needs its resources again, no more resources will be shared to the second group (B) till it gets all of them. In fact, this group (A) receives a compensation and gets more priority for using (B)'s resources although (B) is still needing them.

The maximum number of cores that a group Y may use in a cluster X is named **quota** and is calculated by multiplying the cores they own in that cluster with a **scale factor (SF):**

$$quota_{yx} = \#cores\_group_{yx} \cdot SF$$

Nowadays, the scale factor (SF) chosen is **1.25**.

For example, a group that owns 3 iqtc04 nodes has a total of 36 cores to use (each node has 12 cores, so 3*12=36). With the scale factor, this group may use up to 45 cores of the same cluster. As seen, the scale factor is calculated always for computing nodes of each cluster.

2. Calculating share

The ***share*** that each group has, depends on the number of cores they own. It also may be considered that not all cores have the same value. The cores of the newest cluster must have more value than the old ones.

The way the cores of each cluster are evaluated is calculating a ***weighting*** (w) by comparing each machine with the oldest ones (cerqt2 cluster) and depending on the GFLOPS of performance of each processor.

As a result, these are the weighting values of a processor on each cluster:

weighting of a processor in iqtc01 = 2.375
weighting of a processor in iqtc02 = 7.685
weighting of a processor in iqtc03 = 6.73
weighting of a processor in iqtc04 = 12.4

Notice that each node has 2 processors:

1 node of iqtc01 has 2 Dual-Core processors and so, 4 cores
1 node of iqtc02 has 2 Quad-Core processors and so, 8 cores
1 node of iqtc03 has 2 Quad-Core processors and so, 8 cores
1 node of iqtc04 has 2 Six-Core processors and so, 12 cores

Knowing this, the share of each group and cluster is calculated by the following formula:

where n=4 (the four clusters)

The **share of a group Y** is calculated by the summation, for each cluster X, of the number of cores by processor the group owns in that cluster X multiplied by the weighting (w) of a core in that cluster.

3. Monitoring fair sharing

There are some ways to monitor Fair Sharing. The two best options are the ***check_ocupacio*** command and the web site Cluster Usage by Group.

Other tools are available for IQTC users. A brief explanation of each of them may be found here.

command `check_ocupacio`

It may be executed by a user from access portal and shows the current queues occupation an the percentage of his/her group usage.

```
$ check_ocupacio


TOTAL ACTIVE NODES (iqtc01,iqtc02,iqtc04) 1265/1508
In this script are not included the interactive nodes

List of cores of each cluster (free/used/total)
IQTC01= 88/188/276
IQTC02= 33/191/224
IQTC04= 122/886/1008


CLUSTER    GROUP    USED_CORES/FREE_CORES/TOTAL_CORES        %USAGE    %FREE    %
GLOBAL_USAGE
The group g11 has originally 10 cores, with fairsharing(x1.5) has a maximum of 16
cores

iqtc01        g11        12/4/16              75.00%           25.00%
0.95%

The group g11 has originally 10 cores, with fairsharing(x1.5) has a maximum of 16
cores

iqtc02        g11        0/16/16               0.00%          100.00%
0.00%

The group g11 has originally 12 cores, with fairsharing(x1.5) has a maximum of 18
cores

iqtc04        g11        0/18/18               0.00%          100.00%
0.00%
```

In CERQT2 cluster case, the occupation shown is for user, not by group.

The above table shows a sample output of the check_ocupacio command. In first place shows the free, used and total cores for each cluster. After this, it is shown for each cluster the number of cores the group owns and the total that it is available to use due the scale factor. It also shows how many cores is currently using and how many are still free.

Also, if you pass the name of one of the IQTC clusters as a parameter of check_ocupacio command it will print only the information relative to this cluster.

Cluster usage by group web site

This web interface show the accounting for a requested group in each cluster. The link is:

http://portal.qt.ub.edu:8080/sgenew/qacct_grup.php?view=hourly&g**roup=gx**

Where group=gX must be changed for the number of each group.

For example, for group g1 the link would be:

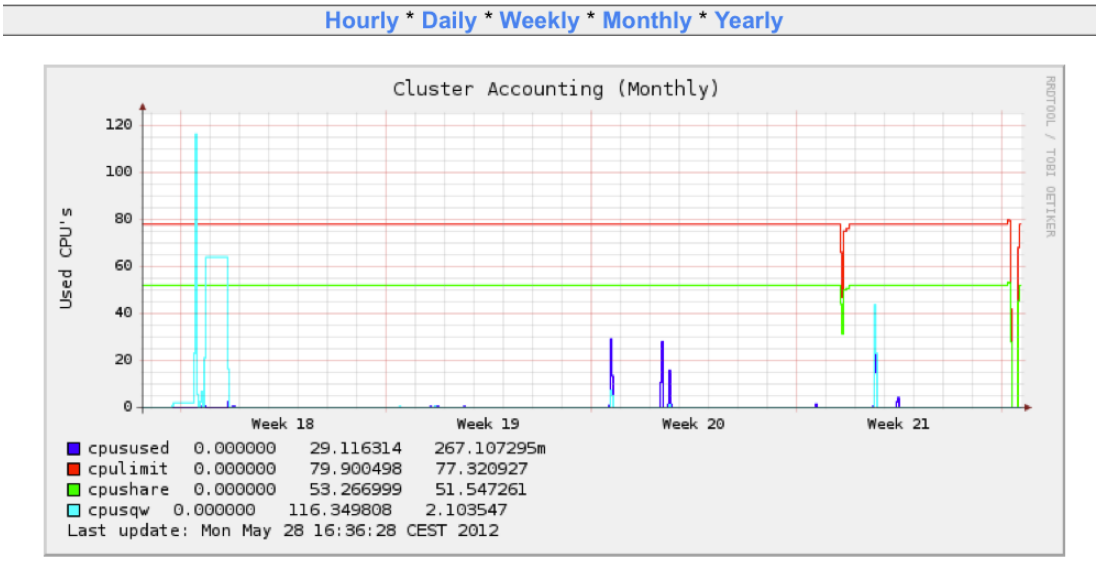
http://portal.qt.ub.edu:8080/sgenew/qacct_grup.php?view=hourly&group=g1

For group g2:

http://portal.qt.ub.edu:8080/sgenew/qacct_grup.php?view=hourly&group=g2

And so on.

The following figure is a sample of the kind of graphics that are available:



Graphics may contain information relative to the last hour, day, month, week or year and it is available by clicking on the corresponding link.

The **horizontal green** line corresponds to the cores the group own.

The **horizontal red** line corresponds to the maximum of cores the group may use thanks to the factor scale.

The **dark blue** lines shows the number of cores used during the time specified.

The **light blue** lines show the number of cores in waiting time.

See more information here.

Submit scripts

1. Submit scripts

A submit script is a script used to send jobs to SGE. It specifies the necessary configuration to be executed (queue, cores, parallel environment, modules, etc.), the input data needed by the application wanted to run and, optionally, technical requirements such as memory, disk, etc.

The submit script also must define where (which directory) the job will run and where the output data and results will be saved.

1.1 Running directories (scratch vs work)

As said previously, once the submit script is submitted, SGE will manage it in order to execute a calculating application with the configuration and data specified.

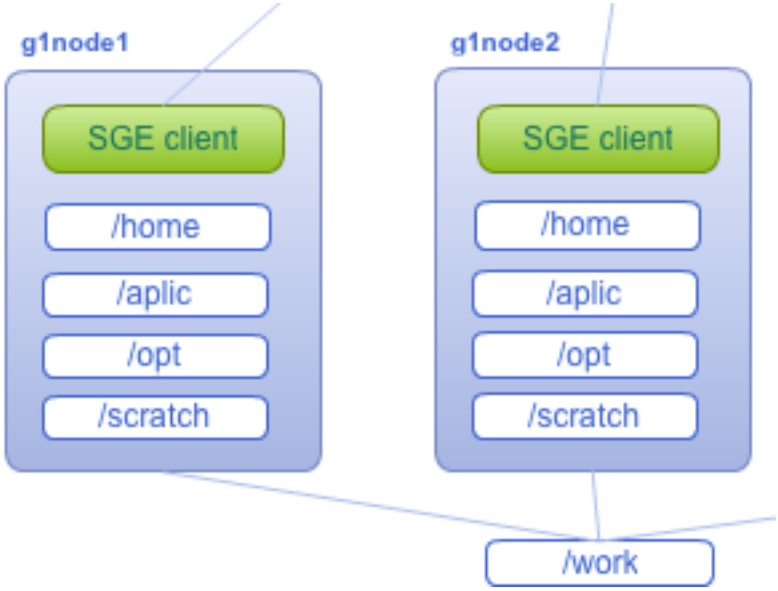
Sometimes the application will require to be executed in a singular computing node, other times parallelization will be required (see IQTC users guide: parallel environment for more information).

In any case, the application must always be able to access to data and logging files. For this reason, two special directories have been reserved for calculating and must be used correctly:

- **/scratch** --- This is a local directory that may be found in every computing node. Every IQTC user has an special directory under /scratch reserved to store their calculating data while the jobs are running. This space is large enough to contain all this data and it is **planned to be used by jobs that does not require to parallelize**. As the job is running inside a unique computing node, it has sense it also use the /scratch directory it owns. Therefore no network resource is used and jobs run efficiently. There is a special environment variable to access to it: **$TMPDIR**

- **/work** --- This is a shared directory between nodes. For this reason, this directory is reserved specially to be used by jobs that need to parallelize. There is a special environment variable to access to it: **$WORK**

The following schema illustrates the differences between scratch and work:



Summary:

if you are using a single computing node -> use /scratch ($TMPDIR)

if you are using more than a computing node -> use /work ($WORK)

1.2 Environment variables

There are several sge variables that may helpful to be used inside a submit script:

$HOME: /home directory for the user that executes the script.

$USER: Username.

$JOB_ID: Identifier number for the job.

$JOB_NAME: job name specified at the submit script.

$TMPDIR: Local directory where smp jobs should work. The content saved here it is removed when the job ends. Remember to add in your submit script some lines in order to copy the necessary results to your home.

$WORK: Shared directory between nodes. The data saved here it is not deleted when the job ends. In order to save space for all users, please add some lines in your submit script in order to delete your data from there before the job ends.

1.3 Submit script structure


Several samples of submits scripts may be found at /scripts directory.

By convention, this script will use the extension **.sub** and be started with the line **#!/bin/bash**

Notice there are several symbols that determine whether a line must be interpreted by SGE or is part of the script to be executed:

#! Followed by /bin/bash is wsed to indicate the file will be a script.

#$ Indicates that the rest of the line is a parameter to be interpreted by SGE.

# Indicates that the rest of the line is a comment.

##$ Indicates it is a comment of a parameter for the SGE.

If there is **no symbol**, indicates that line is part of a script

In the following figure there is a submit script sample that will be explained line by line in the next sub sections.

```
#!/bin/bash
# Basic parameters: job name, parallel environment and cores, queue, used
shell,
# current working directory,output files (.err, .out), email.
#$ -N DL_POLY_SMP_IQTC4
#$ -pe smp 12
#$ -q iqtc04.q
#$ -S /bin/bash
#$ -cwd
#$ -o dl_poly_smp.out
#$ -e dl_poly_smp.err
#$ -m e

#$ -M yourmail@ub.edu

# Load the modules needed
. /etc/profile
module load dl_poly/2.17_ompi

# Copy inputs and files needed to the directory where the jobs will run
cp -r /home/iqtc/inputs/dlpoly/N14_LONG_NP8_IQTC04/* $TMPDIR/
cd  $TMPDIR

# Run the job
export OMP_NUM_THREADS=$NSLOTS
DLPOLY.X


# Copy the results to our home directory
mkdir -p $HOME/TESTS/DLPOLY2.17/$JOB_NAME/$JOB_ID
cp -r * $HOME/TESTS/DLPOLY2.17/$JOB_NAME/$JOB_ID
```

- ***Configuration***

1. Create a file with .sub as extension

2. Edit the file and indicate that the file will be a script. Notice this line is **mandatory** and it does not matter if you are going to use bash or tcsh. This line must be as follows:

```
#!/bin/bash
```

3. Indicate the script type: bash or tcsh

If using bash:

```
#$ -S /bin/bash
```

If using tcsh:

```
#$ -S /bin/tcsh
```

4. Indicate script name

```
#$ -N DL_POLY_SMP_IQTC4
```

5. Indicate number of cores and parallel environment (smp/omp*)

```
#$ -pe smp 12
```

In this example, the user is asking for smp as parallel environment and 12 cores.

6. Indicate the queue where the job must be sent. You can check queues names here.

```
#$ -q iqtc04.q
```
In this example, the queue selected is the one corresponding to iqtc04 cluster.

7. Indicate that the script is running from the current working directory.

```
#$ -cwd
```

8. Indicate a name of the files where the script will log the output (see flag -o) and possible errors (see flag -e).

```
#$ -o dl_poly_smp.out
#$ -e dl_poly_smp.err
```

9. If it is wanted an e-mail to be sent when the job finishes, indicate an e-mail address.

```
#$ -m e
#$ -M yourmail@ub.edu
```

The option -m e tells SGE that a mail is needed when the job finishes.

The option -M specifies the e-mail address.

- ***Modules selection***

In this part of the script, user must specify all modules needed to run the job. In order to understand the

following module command, it is required to write first this line:

```
. /etc/profile
```

The module that corresponds to each application may be consulted here. This is an example of how would be loaded the module for the version 2.17 of dl_poly application planned to be used with parallellization.

```
module load dl_poly/2.17_ompi
```

- ***Input section***

Most applications need a special input data in order to be executed. Each application will require a different data as input, therefore it will be necessary to consult application documentation to see how to write a correct input.

In the submit script will be necessary to copy all input and other necessary files to the directory where the job will be running.

```
cp -r /home/iqtc/inputs/dlpoly/N14_LONG_NP8_IQTC04/* $TMPDIR/
cd  $TMPDIR
```

The above sample is smp, for this reason copy all necessary data to $TMPDIR. Then change to that directory to make all calculating from there.

- ***Execute job***

Each application needs to be called differently. For this reason, is necessary to read its documentation.

```
export OMP_NUM_THREADS=$NSLOTS
DLPOLY.X
```

- ***Copy results and log files***

In this sample, a directory is created to save results in it. Two special system variables are used: $JOB_NAME and $JOB_ID

```
mkdir -p $HOME/TESTS/DLPOLY2.17/$JOB_NAME/$JOB_ID
cp -r * $HOME/TESTS/DLPOLY2.17/$JOB_NAME/$JOB_ID
```

2. Request special resources

In the submit script, some special request may be reserved. They must be preceded by `#$ -l`.

- ***2.1.1. mem_free***

Defines the amount of ram memory necessary **per core** (example: mem_free=2G).

This means that the node or nodes assigned to the job will have, at least, 2GB of memory free by each core requested. In case the job was requesting 12 cores, the node assigned at least will have 24GB of RAM free.

Be careful, the quantity of memory indicated is multiplied by the number of cores. For example, if a job requires 5GB in iqtc02 and is requesting 8 cores, it is really asking 5x8=40GB!! As iqtc02 nodes only have from 6 to 8GB, the job will never have enough resources.

2.1.2. exclusive=true

reserves a complete node for the execution of the job. It means no other jobs will run in that node. It is a good way to assure all memory, cpu and disk space will be reserved for this job. **This resource is only available when using the 'omp*' PE on all the clusters and 'linda' PE on IQTC04.**

For node exclusivity when using the 'smp' PE, please see the Node exclusivity section.

2.1.3. scratch

Defines the amount of space necessary in the /scratch file system.

For example: You know your job will need at least 800GB of free space in /scratch . In most of iqtc04 nodes, /scratch has about 903GB. This means that your job will need almost all the space available in /scratch. If other jobs use the same node as yours, it is very probable that your job ends with a failure because of space problems.

In order to reserve the needed space in scratch, add the following line in your script:

#$ -l scratch=800.000M
Where 800.000M is the free space in /scratch your job is going probably to use.

This is an advantage for your job!! and also for the jobs of others users!

2.1.4. OMP_NUM_THREADS

The OMP_NUM_THREADS is a variable that prevents the OS (Operating System) to parallelize itself an application. This variable is used in a submit script in case of using some applications that require the mpirun command in order to be executed. If you need to control the way the job is parallelized, you may need to set it equals to $NSLOTS or 1.

For example, if you want to run a parallel application with mpirun where you specify "-np n", where n is number_of_cores required, the application will be parallelized n times the application (called n threads). After that, the OS will try to parallelize each thread the same number of cores available in the machine.

For preventing this you should set OMP_NUM_THREADS to 1 before running the application:

export OMP_NUM_THREADS=1 (in case of bash)

setenv OMP_NUM_THREADS 1 (in case of tcsh)

Otherwise, note that we want to set to 1 the *OMP_NUM_THREADS* variable when we want to use the mpirun command, and we're using the ompi parallel environment.

In case of serial applications, so smp parallel environment, you can use the following line:

export OMP_NUM_THREADS=$NSLOTS (in case of bash)

setenv OMP_NUM_THREADS $NSLOTS (in case of tcsh)

In this case we are indicating to OS that we allow its to parallelize the application $NSLOTS times (note that NSLOTS is the number of cores reserved in the job).

Example of sumiting a batch job with the command qsub (see Submit a job section) in iqtc04.q requesting an entire node with 6 cores, 2 GB of memory per core and 100GB of free space in scratch using SMP:
qsub -q iqtc04.q -l mem_free=2G -l scratch=100G -pe smp 6 SCRIPT_NAME

3. Node exclusivity

When using the 'smp' PE, sometimes you may need to use all the memory of a node, but only use one or a few cores. To achieve this, basically you need to request all the cores of a node(4 on iqtc01, 8 on iqtc02, 12 on iqtc04, 64 on iqtc05) and consider the following instructions in your script.

If your application uses OpenMP (MOLCAS, for example), you must have the following line in the script before calling the application:

With tcsh:
```
setenv OMP_NUM_THREADS $NSLOTS
your_super_application
```

With bash:
```
export OMP_NUM_THREADS=$NSLOTS
your_super_application
```

If your application uses MPI:
With tcsh:
```
setenv OMP_NUM_THREADS 1
mpirun -np $NSLOTS your_super_application
```

With bash:
```
export OMP_NUM_THREADS=1
mpirun -np $NSLOTS your_super_application
```

4. Job Restrictions

There are some restrictions to be aware off:

- If a job requires less equal cores than those from a node of the queue where is it being sent, then the use of smp parallel enviroment is mandatory. For example, if the job requires 12 cores in iqtc04.q queue, the parallel environment ompi will not be allowed.

- If a job plans to use the parallel environment linda, it must specify the special resource exclusive=true.

Help command

There is help through command line by typing the command: **man**

Man shows information relative to a command and its parameters, so it is useful to use when you do not remember how exactly a command must be executed or what it really does.

For example,

```
$ man qsub
Reformatting qsub(1), please wait...

SUBMIT(1)                    Grid Engine User
Commands                   SUBMIT(1)

NAME
       qsub   -  submit a batch job to Grid Engine.
```

1. List available queues

To list all available queues, use command: **qconf -sql**

```
$ qconf -sql


cuda.q
g1gpu.q
interactive01.q
interactive02.q
interactive04.q
iqtc01.q
iqtc02.q
iqtc04.q
iqtc05.q iqtc06.q
```

2. List available parallel environments (PE)

To list all available parallel environments, use command: **qconf -spl**

```
$ qconf -spl


adf
gpu
jumbo
lammpi
linda
linda_2
linda_3
make
mpi
mpi_2
mpi_3
ompi
ompi_2
ompi_3
ompi_4a
ompi_4b
ompi_4c
qchem
qchem2
smp
```

* There are several ompi_x parallel environment, one for each set of computing nodes in a rack. But in submit script is enough to specify omp* when a ompi PE is planned to be used.

3. Submit a job

3.1 Submit to a batch queue

Once there is a submit script (for example, script.sub) with all the necessary configuration, the easiest way to submit a job is:

```
qsub script.sub
```

It is possible to specify some configuration at the moment of submitting instead of in the submit script itself. For example, to submit a job into iqtc02.q queue requesting 16 cores and 1GB per core with OMPI as parallel enviroment:

```
qsub -q iqtc02.q -l mem_free=1G -pe omp* 16 script.sub
```

These parameters at the moment of submitting a job, have priority from the ones inside the script. In the example above, no matter if another queue was specified inside the script, the job will be queued to iqtc02.q.

3.2 Submit to an interactive queue

In order to open an interactive session to execute a job, the command **qrsh** must be used. The syntax is:

```
qrsh -q QUEUE_NAME [-l RESOURCE] -pe PE_NAME total_cores
```

where:

QUEUE_NAME: corresponds to one of the available interactive queues. For example, interactive04.q

RESOURCE: is an optional parameter and allows user to specify some resources to have a specific value. See Request special resources section for more information.

PE_NAME: corresponds to indicate the PE to be used. It must be followed by the amount of desired cores for the session. See IQTC users guide: parallel environment for more information.

A sample:

```
qrsh -q interactive02.q  -pe ompi 8
```

 If you want to guarantee that the interactive node is all for you, set the flag -pe smp to the total number of cores of the node. For example, in case of the interactive04.q you could use:

```
qrsh -q interactive04.q  -pe ompi 12
```

 You are able to use an interactive queue **during 12 hours**. After this time, you will be logged out from it.

 In case of iqtc06 cluster there isn't an interactive06.q queue, in this case you must use the same iqtc06.q queue for using a node in interactive mode.

3.3 Submit to a computing node

There are two ways of submitting a job to a singular computing node or to a pool of them:

A) Submit to a node adding ***@node_name*** at the queue name:

```
qsub -q iqtc01.q@g2node4 -pe smp 4 script.sub
```

B) Submit to a pool of nodes that starts with specific characters by adding ***@characters**** at the queue name. Notice the character * to notate that after the characters specified, all are possible. A sample of submitting a job to g1 nodes from iqtc01 queue:

```
qsub -q 'iqtc01.q@g1*' -pe omp* 8 script.sub
```

If there are some nodes to avoid from the pool specified, it may be specified by adding the character ! as follows:

```
qsub -q 'iqtc01.q@!g1node1,iqtc01.q@g1*' -pe omp* 8 script.sub
```

In this example, the node g1node1 from iqtc01.q will be avoided.

C) Users may define a file, with default options for all jobs they submit. This file is "*.sge_request*" and must be placed in user's home directory:

$HOME/.sge_request:
  -soft [OPTIONS] -hard [OPTIONS]

There are two different policies, -soft and -hard:
-soft: defines some parameters that will be satisfied if they are available, if not the job will be executed without satisfying it.

-hard: the job won't run until the specified options are satisfied.

Examples:

```
  -soft -q 'iqtc01.q@g2node10,iqtc01.q@g4node7' -hard
```

Jobs will try to be executed in nodes g2node10 or g4node7, if they are being used or are not available, jobs will be executed in other nodes.

```
  -soft -hard -q 'iqtc01.q@g2node10,iqtc01.q@g4node7'
```

Jobs will be executed in nodes g2node10 or g4node7, if they are being used or are not available jobs will be waiting until those node are available.

In that file just default values will be defined. If users define some other options that are in conflict with those in the file, in "qsub" command or in the script, the .sge_request options will be ignored.

4. Delete a job

Once a job is submitted it is possible to delete it using its job id.

For example, to delete a job with 987 as job id:

```
qdel 987
```

5. Modify a submitted job

The command qalter allows to modify some requested resources for a already submitted job before it is executed (at the waiting time). This command is followed by the corresponding flag of the attribute to change. Some examples:

5.1 Modify parallel environment

```
qalter -pe omp* 4  JOB_ID It changes the parallel environment or the
number of cores
```

5.2 Modify priority

You may change the priority of your sending jobs, but only to decrease it. This may be interesting if you need a sent job to run before than the rest. You may accomplish this by decreasing the priority of the rest of your jobs on queue waiting.

```
qalter -p -1024 JOB_ID
```

Note that you are able to use any range within -1024 to 0 .

6. Hold/restart a job

To let a submitted job into standby mode, use the command: ***qhold***

To restart a job that was previously held, use the command: ***qrls***

7. Show all job status of a user

Once there are job submitted by a user, he/she may check their status using the command: **qstat**

```
$ qstat
job-ID  prior   name       user         state submit/start
at     queue                          slots ja-task-ID
-----------------------------------------------------------------------
-----------------------------------
 185800 0.14940 siesta2j   jingles      r     07/23/2012 08:57:18
iqtc05.q@g1nodea2                32
 185801 0.14940 siesta2j   jingles      qw    07/23/2012
09:02:48                                   16
 187754 0.14940 h2f2       jingles      r     07/30/2012 12:06:54
iqtc02.q@g4nodei4                 4
```

As seen in the above table, this user has three jobs submitted. For each of them, it shows its job id, its priority, the submit script name, the user name, the state, the date and hour when it was submitted or when it started, the queue and the master node assigned to it and the number of cores it requires.

The state may have different values:

qw - The job is waiting to be assigned available resources

r - The job is currently being executed

8. Show all jobs by cluster

In order to see all the running and queued jobs by a cluster you may use the following commands:

- check_iqtc01
- check_iqtc02
- check_iqtc04

9. Show details of a submitted job

Once a job is submitted some relevant information may be checked by using command: qstat -j JOB_ID

For example:

```
~$ qstat -j 187754
==============================================================
job_number:                 187754
exec_file:                  job_scripts/187754
submission_time:            Mon Jul 30 12:06:37 2012
owner:                      jingles
uid:                        1015
group:                      iqtc
gid:                        10000
sge_o_home:                 /home/jingles
sge_o_log_name:             jingles
sge_o_path:                 /home/jingles/bin:/usr/local/git-
1.7.2.1/bin:/sge/bin/lx24-
amd64:/usr/local/bin:/usr/bin:/bin:/usr/games:/usr/java/jdk1.6.0_15/bi
n
sge_o_shell:                /bin/bash
sge_o_workdir:              /home/jingles/G09
sge_o_host:                 portal02
account:                    sge
cwd:                        /home/jingles/G09
stderr_path_list:           NONE:NONE:$HOME/G09/h2f2.err
reserve:                    y
mail_options:               e
mail_list:                  administracion.iqtc@qi.ub.es
notify:                     FALSE
job_name:                   h2f2
stdout_path_list:           NONE:NONE:$HOME/G09/h2f2.log
jobshare:                   0
```

```
hard_queue_list:              iqtc02.q
shell_list:                   NONE:/bin/csh
env_list:
script_file:                  h2f2
parallel environment:  smp range: 4
project:                      iqtc
usage    1:                   cpu=23:37:57, mem=0.00000 GBs, io=0.00000,
vmem=N/A, maxvmem=N/A
scheduling info:              queue instance "iqtc01.q@g10node2" dropped
because it is temporarily not available
```

- check_iqtc05

10. Check queues status

With the command qstat -f it is available to see the general status of all queues:

```
~$ qstat -f

queuename                     qtype resv/used/tot. load_avg
arch          states
---------------------------------------------------------------------------
------------
cuda.q@g6gpu01                BIP   0/1/1          1.00     lx24-
amd64
---------------------------------------------------------------------------
------------
cuda.q@gpu01                  BIP   0/1/1          1.01     lx24-
amd64
---------------------------------------------------------------------------
------------
g1gpu.q@g1gpu01               BIP   0/0/12         0.00     lx24-
amd64
---------------------------------------------------------------------------
------------
interactive01.q@g9node1       IP    0/0/4          0.00     lx24-
amd64
---------------------------------------------------------------------------
------------
interactive01.q@g9node2       IP    0/0/4          0.00     lx24-
amd64
---------------------------------------------------------------------------
------------
interactive02.q@g1nodei1      IP    0/0/8          0.00     lx24-
amd64
---------------------------------------------------------------------------
------------
interactive02.q@g9nodei5      IP    0/0/8          0.00     lx24-
amd64
---------------------------------------------------------------------------
------------
interactive04.q@g9noden4      IP    0/0/12         0.00     lx24-
amd64    E
---------------------------------------------------------------------------
------------
interactive04.q@g9noden5      IP    0/0/12         0.00     lx24-
```

```
amd64
---------------------------------------------------------------------------------------------
---------------
iqtc01.q@g10node1               BP    0/0/4          0.00     lx24-
amd64   ---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node2               BP    0/0/4          -NA-     lx24-
amd64    au
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node3               BP    0/0/4          0.00     lx24-
amd64    d
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node4               BP    0/4/4          4.02     lx24-
amd64
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node5               BP    0/4/4          4.02     lx24-
amd64
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node6               BP    0/4/4          4.04     lx24-
amd64
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g10node7               BP    0/4/4          4.01     lx24-
amd64
---------------------------------------------------------------------------------
--------------------
iqtc01.q@g11node1               BP    0/4/4          0.21     lx24-
amd64
---------------------------------------------------------------------------------
--------------------
```

As shown in the table above, there are several columns. The most important to check are the following:

resv/used/tot. It shows how many cores of this queue are reserved, used and in total. For example, 0/4/4 mean there are no reserved cores for this user, there are 4 cores currently used by one or more jobs and the total of cores of this queue is 4.

load_avg. It shows the load average for a queue. It normally shows 1 unit of load average for core used. If there is more than that, means that the node is over loaded probably because a job (or more) is using more resources than than the ones available.

states. Shows alerts,errors, etc. for the queue.

d - means the node is disabled. Probably because of maintenance.

au - means that the SGE is not available in that node. The node has some kind of problem.

E - means the queue has a temporary error