CONNECTING

Once you have obtained an user account (those users who already have an active account in Superdome and HPC320 servers have the same account in this system), you can connect to the SVG system with the login and password data. The name of the server is svgd.cesga.es and the connection mode is made by an ssh client. A client for Windows (putty) can be found in this link, in case of using Linux you can use the ssh command. Files can be transferred towards and from the SVG using a Windows client as WinSCP or the Linux scp command. You must bear in mind that the SVG has a firewall that restricts the connections from and to external servers.
In order to visualize the graphical output of programs that have this possibility, you must consult the FAQ section.

INTERACTIVE USE

The SVG operating system is Linux (based on red Hat Entreprise 4.0). Once inside the system, an interactive session will be opened through a shell that will be bash by default. This shell has some CPU time, memory and disk limits imposed that can be consulted by the command ulimit -a (0,5 CPU hours and 512MB of memory at the moment). The use of the queue system is recommended for the execution of jobs as well as the use of the compilation environments in order to compile applications or to run interactive jobs.

BATCH SYSTEM

For jobs that require more resources (CPU time, memory or space on disk) than those limited by the interactive shell, the batch system or the compilation environments must be used. This system is Sun Grid Engine. The way to send jobs implies knowing beforehand the maximum values of time quantity, number of processors, architecture, memory and space on disk that the job is going to require. This also allows a better use of the system resources by all the users.

The command to send jobs to the batch system in the SVG is qsub, followed by a list of the resources the job needs. For instance, let us suppose that we want to send a Gaussian job that does not require more than 2 Gigabytes of memory and 10 Gigabytes of scratch, and we estimate an approximated execution time of no more than 24 hours, using one processor. If the entry file for Gaussian is called script.com and it can be found in the “pruebas” directory, the way to send this job to queue would be:

```
qsub -l num_proc=1,s_rt=24:00:00,s_vmem=2G,h_fsize=10G
cd $HOME/pruebas
g98 < script.com
control+D
```

If there is no mistake, we will get this kind of message:

```
Your job 492 ("STDIN") has been submitted
```

The number 492 is what is called job identifier or JOBID and it allows us to identify our job within the batch system (for instance, in order to use the command qstat and to know whether it is being executed, in-queue, etc. or not). It is also important to indicate this number in consults by phone or e-mail (see the section Support).

The way to specify resources is with the option -l, followed by the resources that are being required separated with a “,”. It is essential to leave a blank space between the option “-l” and the resource list. The

resources must be:

Resource	Meaning	Units	Minimum value	Maximum value
num_proc	Number of CPUs (processors) required by the job	Whole number	1	16
s_rt	Real maximum time a job can last for	TIME	00:01:00	300:00:00
s_vmem	Total amount of RAM memory required for the job	SIZE	112M	4G
h_fsize	Maximum space required by a single file created by the job	SIZE	1M	120G
arch	Type of processor in which the job is desired to be executed Possible values: 32, 64, opteron, bw			

If you need to use superior values to the limits of these resources or if you need a priorization of your jobs, consult the page of Special Resources, there the steps to be followed are indicated.

The format of the units for the resources is the following:

TIME: it specifies the maximum time period during which a resource can be used. The format is the following: [[hours:]minutes:]seconds, for example:
00:30:00 are 30 minutes
100:00:00 are 100 hours
1200 are 1200 seconds (20 minutes)

SIZE: it specifies the maximum size in Megabytes. It is expressed in the form whole[suffix]. The suffix acts as a multiplier defined in the following table:
K Kilo (1024) bytes
M Mega (1,048,576) bytes
G Giga (1,073,741,824) bytes

The nodes available in the SVG at the moment correspond to the following kinds of machine:

	Intel 32 bits	Intel 64 bits	Opteron 64 bits
num_proc	1	1	4
s_vmem	1G/2G	2G	4G
s_rt	300:00:00	300:00:00	300:00:00
h_fsize	120G	120G	120G
arch	32 bits	32 bits	64 bits (opteron)

All the previous nodes have a Gigabit network and they allow the sending of parallel jobs using the aforementioned network. PIII nodes with Myrinet network are also available, but these are reserved for the realization of parallel jobs aiming to the realization of tests or the optimization of parallel algorithms.

We must bear in mind that:

It is very important to leave a blank space between the option “-l” and the following resource, while there should not be any other blank space separating the resources after that.

These values are maximum, so they cannot be exceeded. This means that if we believe that our job will last around 23 hours, we should put s_rt=24:00:00 in order to be sure of leaving a safety margin. After 24 hours the system will finish the job automatically, even though it would not have been finished.

The more these values are in keeping with the resources the job actually consumes, the more priority the job will have.

If these resources are not enough for the job, it will be aborted due to lack of resources and it will be necessary to indicate the proper values. In general, we recommend applying for the closest resources, above the resources estimated. The reason is that the less resources are applied for, the more priority the job will have and the sooner it will run.

To sum up, let us show some examples for different jobs:

1. For a job that requires few memory consumption and execution time:
qsub -l num_proc=1,s_rt=00:10:00,s_vmem=200M,h_fsize=100M job.sh

2. A job that requires much execution time (80 hours) and few memory (256Mb is enough)
qsub -l num_proc=1,s_rt=80:00:00,s_vmem=256M,h_fsize=10M job.sh

3. A job with great memory requirements (4GByte) but few execution time:
qsub -l num_proc=1,s_rt=00:30:00,s_vmem=4G,h_fsize=10M job.sh

4. A job that generates a big result file (up to 20Gigabytes):
qsub -l num_proc=1,s_rt=30:00:00,s_vmem=500M,h_fsize=20G job.sh

5. A job that consumes 100 CPU hours, 2 Gigabytes of memory and generates a 5-Gigabyte file:

qsub -l num_proc=1,s_rt=100:00:00,s_vmem=2G,h_fsize=5G job.sh

6. A parallel job with 8 processors and 10 hours of total execution time, 8Gigabytes of total memory and which generates a 10-Gigabyte file:
qsub -l num_proc=8,s_rt=10:00:00,s_vmem=8G,h_fsize=10G job.sh

If you need to use values superior to the limits of these resources, you must make an application as it is explained in the corresponding FAQ.

Once we execute the command qsub and we obtain the identifier for the job, it passes to an appropriate queue for its execution. The job will wait in turn for the moment when the required resources are available, to go to execution, and finally the job will finish and it will disappear from the queue.

Checking the state of the jobs:

In order to check the state in which the jobs are, the command qstat can be used. We will obtain an output as the following one:

job-ID prior name user state submit/start at queue slots ja-task-ID

1134360 2.98007 mm5-11.sh orballo r 10/26/2006 08:19:09 normal@compute-1-27.local

The meanings of the fields are the following:

Job-ID: 489 is the value of the JOB-ID that was assigned to the SGE queue system. The JobID is a unique identifier valid for every job and it allows the monitoring of it.

Prior: indicates the priority with which the job is being executed.

Name: STDIN is the name of the job that was sent to queue. If a job was sent from the standard input (that is, writing the commands intensively when the job was sent), STDIN will appear. In the event of being a script, the name of the script

will appear.

User: orballo is the login of the user who sent the job to queue

State: “r” is the state in which the job is at the moment, and it indicates that it is running. The other possible states of a job are:
t: transferring the job in order to start running
s: temporally suspended in order to execute other foreground jobs
w: the job is in-queue, waiting for the necessary resources for it to run to be available, or because the user exceeded the limits.Submit/start at: date and hour when the job was sent to queue or when it started running

Queue: normal@compute-1-21.local is the name of the queue to which the job was sent. The first part indicates the name of the cluster queue (normal), and the second part the node where the job is running (compute-1-27.local). The destination queue will depend on the resources requested.

Slots: indicates the number of nodes where the job is running. Normally, this number is 1 for sequential jobs and more than 1 for parallel jobs. In case of parallel jobs the command qstat -t can be used in order to see the rest of the nodes where the job is running.

FILE SYSTEMS

There are different file systems with different characteristics depending on the space and access speed requirements.

Home directory

It is the directory where the habitual daily work data and files will be, of which backups are made regularly. There are quotas (limits in its utilization), so its use must be moderated.

Scratch directory

It is a storage space for temporal data used in applications such as Gaussian or Gamess that require a big file where a great amount of data is written continually. It is only possible to access this directory through the queue system and its name is $TMPDIR. The data that can be found in this directory will disappear when the job is finished. If any file contained in this directory was necessary, it is every user's responsibility to copy it to their home directory before finishing the job or to specify the option -v COPIA=$HOME/destino when the job is launched with the qsub (see FAQ for more details).

/tmp directory

In this access directory common to all users small temporal files can be introduced, although their utilization is not advisable, and their content can be eliminated periodically.

Parallel file system

It is an information space common to all the cluster nodes and its utilization is advisable for jobs that require massive information processing (data-mining, etc...). In order to make use of this parallel file system, you must follow the instructions given in the storage services guide.

Shared directory

It is a storage space visible from all the CESGA servers. Its utilization is advisable in order to share data among different servers or for calculus requiring shared information. In order to make use of these storage system, you must follow the instructions given in the storage services guide.
COMPILATION ENVIRONMENTS

Many of our users use the front-end of the SVG to compile their programs before sending them to queue. This can be a problem not only for the consequent saturation of the front-end but also for the user, for it is going to be compiled in a 64-bit machine by default, and the program is going to be probably executed in a 32-bit node after being sent to queue.
In order to help solving this problem and to prevent the user from facing the problem of having an application not running properly in the nodes after compiling it or even showing wrong results, the compilation environments are at the user's disposal. They allow generating an environment appropriate for their compiling needs.
In order to open a compilation session it is necessary to execute the following command:
compilar -arch <arquitectura>
where <arquitectura> can be:

32: 32-bit architecture x86_32
64: 64-bit architecture x86_64
opteron: Opteron
bw: BW nodes with Myrinet network

It must be taken into account that each compilation session is limited to 30 minutes.
Anyway, should you have any doubt about the compilation or the use of this script, you can contact us in sistemas@cesga.es.

Availbale compilers and options

The GNU compilers are available in order to compile in 32 or 64 bits: gcc, g++, g77, gcc4, g++4, gfortran (depending on the compilation environment selected, the 32 or 64-bit versions will be used)

Apart from the GNU compilers, the Portland Group compilers are available, which include support for SSE instructions:

For 32-bit environment: 6.1-3 version: only Fortran (pgf77, pgf90, pgf95 and pghpf)

For 64-bit environment: 7.1-2 version: only Fortran (pgf77, pgf90, pgf95 and pghpf)

A specific compiler for Opteron is available, for its utilization it is necessary to contact aplicaciones@cesga.es.

In the case of the Portland 32-bit compilers, it must be taken into account that the 6.1-3 version Fortran compilers are used by default (pgf77, pgf90, pgf95 and pghpf), but the 6.1-1 version is also available.

The optimization option recommended for the Portland compilers is -fast, which selects a set of options intended to optimize the code. As any other optimization option, attention must be paid to the results obtained with the code and it must be checked that they are correct before using it in definitive computing.

The Portland compilers manuals can be consulted in Portland's web page

USING MPI

MPI is a parallel programming interface for the explicit sending of messages among parallel processes (for which it is necessary to have added the MPI code to the program). In order to enable the MPI utilization in the programs, it is necessary to include the MPI head file in the source code and to link to the MPI libraries. In addition, it is not possible to use MPI interactive codes, but it is compulsory to use the queue system of these programs.

Most of the nodes of the SVG cluster have a Gigabit network and allow the sending of parallel jobs using the aforementioned network. PIII nodes with Myrinet network are also available, but these are reserved for the making of parallel calculus that have as an objective the realization of tests or the optimization of parallel algorithms.

User's guide according to necessities

There are different MPI versions available:

MP1 V1 over Gigabit network
MP1 V1 over Myrinet network
OpenMPI over Gigabit

How to use each one is explained below:

MPICH GIGABIT

Add to ~/.bashrc

```
# mpich_p4

export PATH=/opt/cesga/mpich-1.2.7p1_p4/bin:$PAT
```

The job script must specify the following mpirun options:

```
#!/bin/bash

export PATH=/opt/cesga/mpich-1.2.7p1_p4/bin:$PATH
```

export P4_GLOBMEMSIZE=67303416

export P4_SOCKBUFSIZE=65535

mpirun -np $NSLOTS -machinefile $TMPDIR/machines /home/usuario/programa

If the P4GLOBMEMSIZE value is too small, the application stops, showing a error message and suggesting a greater value.

Send the jobs to queue with options similar to these ones:

qsub -cwd -l num_proc=1,s_rt=00:05:00,s_vmem=128M,h_fsize=1G,arch=32 -pe gigabit 4 test.sh

OPENMPI GIGABIT

Add to ~/.bashrc

OpenMPI

export PATH=/opt/cesga/openmpi/bin:$PATH

The job script must specify the following mpirun options:

#!/bin/bash

Set the environment to the appropiate MPI version

openmpi

export PATH=/opt/cesga/openmpi/bin:$PATH

It is important to give the full path to the program

mpirun -np $NSLOTS -machinefile $TMPDIR/machines /home/usuario/programa

Send the jobs to queue with options similar to these ones:

qsub -cwd -l num_proc=1,s_rt=00:05:00,s_vmem=128M,h_fsize=1G,arch=32 -pe gigabit 4 programa.sh

MPICH MYRINET

The job script must specify the following mpirun options:

#!/bin/bash

It is important to give the full path to the program

mpirun -np $NSLOTS /home/usuario/programa

Send the jobs to queue with options similar to these ones:

qsub -cwd -l num_proc=1,s_rt=00:05:00,s_vmem=128M,h_fsize=1G,arch=bw -pe mpi 4 programa.sh

Compilation and linking

For the Fortran codes, it is necessary to include the following directive in the source code of any code using MPI:

INCLUDE 'mpif.h'

and compile with the following command:

mpif77 miprograma.f -o miprograma.exe

For the codes in C, it is necessary to use the following directive:

#include <mpi.h>

and compile with a command similar to the following one:

mpicc miprograma.c -o miprograma.exe

In order to compile using Pgf90:

mpif77 -fc=”pgf90 -tp p7 -Msecond_underscore /opt/cesga/mpich-1.2.7p1_p4/cesga/farg.o” miprograma.f -o miprograma.exe

It must be taken into account that it is only possible to use Pgf90 through the 32-bit compilation environment:

compilar -arch 32

In case of having any doubt, consult the guide for the utilization of the queue system.

USING MATLAB

Matlab normally runs from the queues, but in order to prevent the jobs using Matlab form failing when coming into operation due to lack of licenses, we have implemented a new resource for the qsub command: “matlab”.

The way of use will be adding “,matlab=1” to the habitual qsub line at the end.

This way we are asking for a free license to execute our job.

If the license is not available at that moment, the job will not come into operation until some one gets free.

SUPPORT

How to ask for help:
sistemas@cesga.es for every aspect related to the calculus servers, queue systems, storage, etc ...
aplicaciones@cesga.es in case of consults related to the applications use, requests for new applications or for help with the applications compilation, etc ...