SUPERDOME User Guide

- Last Updated (04.08.2009)

Connecting

Once you obtain an user account (those users who already have an active account in HPC4500 or HPC320 servers already have an account in this system), with the log-in data and password, you can access the Superdome system. Server name is sd.cesga.es and to connect to it you need a ssh client. A client for Windows (Putty) can be found here. Using this same protocol you can send or get files to and from Superdome with a client like WinSCP. To visualize the graphic output that some programs may have, you need to connect to it using the ssh -X command. This will automatically set the environment variable to DISPLAY in the computer you are connecting from. To active this option using the Putty client, you need to go to the Connection-> SSH ->Tunnels menu and toggle the “Enable X11 forwarding” option. A X-Windows client like the X-Win32 installed and running is required .

Interactive Use

The Superdome operating system is UNIX(HP-UX 11iv2). Once you connect, an interactive session will be open through a shell thay can be either ksh or bash. This shell has some time, CPU, memory and disk limitations that can be viewed by typing ulimit -a. For high computing demanding jobs, the batch queue system must be used.

Batch queue system

The batch queue system must be used for jobs that require more resources (computing time, memory or disk space) that those limited by the interactive shell. This system is the Sun Grid Engine. At the time of sending work, it is best to know beforehand the maximum time values, number of processors, memory and disk space that the task is going to use. This leads to a better use of system resources by all the users.
The command to send work to the batch queue system in the HPC320 is qsub, followed by the list of resources that the job needs. For example, supposing that we want to send a Gaussian job that doesn't need for than 2 GB of Memory and 10Gb of scratch. We also estimate that its execution time is not longer than 24 hours, using one processor. If the entry file for Gaussian is called script.com and it is found in the “pruebas” directory. The correct way to send this work is:

```
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 no error is produced, we obtain the following type of message:

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

Number 492 is the job id (JOBID) and it allows to identify our job within the batch queue system (for example, using the command qstat to know if it is running, queued, etc.) It is also important to indicate this number when doing checks via telephone or via e-mail with the systems personnel.

The specification of resources is made with the parameter -l, followed by the resources that are requested separated by a “,”. A blank space is required between the option “-L” and the list of resources. The resources must be:

Resource	Meaning

Ammount

Minimum value

Maximum valeu

num_proc
Number of CPUs (processor) required by the job

1
16

s_rt
Maximum real time that the job may take
Time

300:00:00/num_proc
1000:00:00 (only for num_proc=1)

s_vmem
Total amount of RAM memory required for the job
Size
 550M
64G

h_fsize
Maximum disk space required by a single file created by the job.
Size

100G

If you need higher values to these resources limits or a prioritization of your jobs check the website Special Resources. There you will find all the information.
We must bear in mind that:

It is very important to leave a blank space between the option “-l” and the next resource. After that there must not be blank spaces between resources.

1.These values are maximum, so they cannot be exceeded. This means that if we believe that our job will take 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, whether it is finished or not.

The more these values fit to the specified resources to what the job actually consumes, the more priority will have the job to run.

2. The more these values fit to the specified resources that the job actually needs, it will have more priority to run.
If you need to use superior values to the limits of these resources or if you need a prioritization 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:

If the specified resources are not enough for the job, it will quit and you will need to indicate the right values again. All in all, we recommend to specify resources as close as possible but also a bit over the values deemed necessary. The

reason is that the lesser resources are requested, the job will have more priority in execution.

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

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=10:00,s_vmem=100M,h_fsize=100M trabajo.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 trabajo.sh

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

If you need to use values superior to the limits of these resources, you must request access to the special queue, by sending an email to sistemas@cesga.es

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.
qstat

We will obtain the following output:

Job id
prior
name
user
state
submit/start
at
Queue

489
0
STDIN
carlosf
r
12/29/2003
19:49:05
Cola1

The meaning of the fields is the following:

Job ID: 489 is the JOB-ID value allocated by the PBS batch queue system. JobID is a unique identifier for each job thereby allowing to monitor it.

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

Name: STDIN is the Job name sent to the queue. If the standard entry has been used (IE, by typing the commands when sending the job), STDIN will be displayed. If it is a script, the script name will be displayed.

User: carlosf is the user log-in sent to the queue.
State: "r" is the state the job is in and it also displays this message (running). The other alternative states are:
t: job is being sent to start execution. R shows the job is already running.
s: temporally suspended to run jobs with higher priority.
w: the job is in the queue until enough resources are available to be executed or the resources specified by the user have been exceeded.

Submit/start at: Date and Time the job was sent to the queue or started running.
Queue: queue 1 is the name of the queue the job was sent to. The target queue will vary according to the requested resources.

Master: shows the host the job was sent from.

Parallel jobs execution

Integrity Superdome Cluster is composed of two nodes of 64 CPUs each. Therefore, it is possible to use parallelization models of shared memory (like OpenMP), as models of distributed memory (like MPI) for communication between nodes and within a single node. In both cases, the maximum number of CPUs that can be used will be limited to 16 and it must be always used within the same node.

File systems

Different file systems are available with different characteristics according to the requirements of space and access speed.

Home directory

It is the directory where the common daily work data and files will be, of which backups are made regularly. The amount of data per user in his or her home directory is limited to 10 Gbytes. Using the “quota -v” command the user can know the amount of data that he/she can still put into his or her account at any time.

Storage Systems

Those users who need higher storage demands to the one available in their home directory must request access to the massive storage system, by sending an email to
specifying the storage amount needed and its purpose.

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 batch queue system

and its variable 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 the user's responsibility to copy it to their home directory before finishing the job.

Using Scratch

A directory is created in the local scratch disk of the nodes for every sent job to the queue. This scratch disk is faster than the disks of the users' home directory. Thus, those programs that require many input-output operations should use this scratch directory to work. The name of this directory varies in each execution and can be accessed through the environment variable $TMPDIR within the job. It should be noted that this directory is eliminated when the execution of the jobs is finished. Any important file should be copied to the user's home directory.

Note for Gaussian98 users:

The installation of Gaussian98 in CESGA is configured to use automatically this directory as a scratch space. Therefore, the user does not need to configure the environment variable GAUSS_SCRDIR.

/tmp directory

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

Text Editors

In addition to vi, emacs, xemacs and xedit are available.
Compilation

C/C++ compiler.

There are three C/C++ compilers:

cc, compiles code in C
c89, compiles code in ANSI C
aCC -o CC, compiles code in C++.

By default, the compilation will generate 32bit code. To compile 64bit code it is necessary to include the option +DD64. If you want to produce optimized code for Itanium2, you have to add the option +DSItanium2.

All compilers admit up to four optimization levels that can be activated with the option +Onivel, being level 0, 1(by default), 2, 3, or 4. The -fast or -Ofast optimization selects a combination of options for an optimized execution in speed (specifically, +O2, +Onolimit, +Olibcalls, +Ofltacc=relaxed, +DSnative, +FPD, -Wl,+pi,4M, -Wl,+pd,4M and -Wl,+mergeseg).

All C/C++ compilers support OpenMP 2.0 (see below on how to activate it)

Fortran compilers.
The compiler is the original owned by HP . The main characteristics are:

· Support of F77/F90/F95
· Support of OpenMP 2.0
· Auto parallelizer
· 32bit code generation (by default) or 64 bits

To compile a program you have to type the following command:

f90 fichero.F

In this case, a 32bit code will be generated by default. If you wish to compile for 64bits, you have to include the +DD64 option. If you want to produce specific code for Itanium2 (recommended option) you have to use the +DSItanium2 option or +DSnative option. For example, if you want to compile for Itanium2 with 64bit suppor type the following command:

f90 +DD64 +DSitanium2

Optimization

There are 4 optimization levels that are toggled with +O<nivel> ó -O<nivel>, as well as the possibility of managing compilation options with +O<opcion>. HP Fortran also includes an appropriate optimization for many programs which can be activated with +Ofast. Apart from performing the optimization, it also changes the page of data and program to 4MB, thereby allowing to reduce the loss of TLBs in programs that consume memory or programs that have a large execution file. These paramaters can be changed afterwards by using the chatr command.

f90 +Ofast +DD64 entrada.F

Resizing the execution page

Itanium2 processor includes 128 entries for TLB of type TEXT (program code) and 128 for DATA. It is possible to change the size of the page that the program will use to optimize its performance. The way to do this is through the chatr command and the following syntax:

chatr +pd <page of data size> +pi <execution page size> executable

The valid sizes are 4K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M, 1G, 4G, D, and L, where D stands for "by default" and L for "the largest possible".

For example, suppose we have to execute a program that needs 16GB of RAM memory for data. To distribute those 16GB in only 128 page, we would need 128MB pages, which cannot be selected. Pages of 256MB could be used, in this case, to improve the program performance, using the command:

chatr +pd 256M <executable>

Using MLIB (BLAS/LAPACK/ScaLAPACK)

HP MLIB contains robust and optimized versions of all BLAS functions 1, 2, and 3, BLAS sparse, LAPACK, ScaLAPACCK, SuperLU_DIST, Fast Fourier transforms (FFTs) and convolutions for Itanium2. It also includes the whole METIS 4.0.1 library (http://www-users.cs.umn.edu/~karypis/metis/metis/)

To compile a program that uses these libraries, you have to include the following options lveclib -llapack. The necessary references will be added to the corresponding libraries in each case depending on the data model selected (32 or 64 bits)

For example, the command:

f90 +DD64 entrada.F -lveclib -llapack

It will produce and exec file that will load the libraries (providing they are used in the program), /opt/mlib/lib/hpux64/libveclib.so y /opt/mlib/lib/hpux64/liblapack.so

NOTE: for 64 bits, and only when integers are 64 bits – Integer*8 in Fortran and long long in C, you can include the veclib8 and lapack8 libraries. It is recommended to compile these libraries in a static mode as their performance is better, this way.

Compiling with MPI

HP MPI is based on the implementations of the MPICH standard (from Argonne National Laboratory) and LAM (from University of Notre Dame). It is totally compatible with the MPI 1.2 standard and also includes some of the functions of MPI 2.0, like ROMIO, an MPI/O implementation of ANL. For a comprehensive description of the MPI 2.0 supported features, see the HP MPI User's Guide, Seventh Edition.

There are three commands that allow us to easily compile a program that requires MPI:
mpicc, which by default uses the /opt/ansic/bin/cc compiler
mpiCC, which uses /opt/aCC/bin/aCC
mpif90, which uses /opt/fortran90/bin/f90

Each of them will call the right compiler in each case, with the compiler options specified and it will also include the MPI necessary libraries. As in previous cases, you must also specify whether you are compiling for 32 or 64 bits.

To execute the application you can use the following commands:

For example,
 mpirun -np 4 mpihello

Compilation and execution with OpenMP

Both Fortran and C compiler support the second version of OpenMP. To compile a program that uses OpenMP you must include the +Oopenmp option. If the code includes directives from OpenMP and you still do not want to use it, you must include the option +Onoopenmp. For example, to compile a program.F with OpenMP use the following command:

f90 +Oopenmp -o program program.F

By default, the number of threads initiated equals the number of processors (IE, in the case of Superdome 64). To control the number of threads being executed, use the OMP_NUM_THREADS=<number of threads> variable. For example, under a ksh shell the following command will limit the number of threads to 16:

export OMP_NUM_THREADS=16

If its executed under batch, that variable will have a value equal to the number of processors specified in qsub with the option -l num_proc.

Additional Documentation
For more information and programming manuals, you can visit the following website
http://doc.cesga.es/superdome/Programming.html Here you can find specific information on the Itanium processor and information for the optimization of applications for this processor.