# HPC320 User guide
- Last Updated (03.08.2009)



## Connecting

Once you have obtained an user account (those users who already have an active account in HPC4500 servers have the same account in this system), you can connect to the HPC320 system with the log-in and password. Server name is sc.cesga.es and the recommended access mode is ssh. To use different clients for different platforms see www.openssh.org. A Windows client can be found here, and you can also use a ftp client to send files.

## Interactive Use

The HPC320 operating system is UNIX (Tru64 V5.1A). Once inside the system, an interactive session will be opened through a shell that will be in ksh by default. This shell has some CPU time, memory and disk limits imposed that can be consulted by the command 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:

Recurso

Significado

Unidades

Valor mínimo

Valor máximo

num_proc
Número de CPUs (procesadores) requeridos polo traballo
---
1
4

s_rt
Máxima cantidade de tempo real que pode durar un traballo
Tempo

360:00:00

s_vmem
Cantidade total de memoria RAM requerida polo traballo
Tamaño

8G

h_fsize
Máximo espazo requerido por un único ficheiro creado polo traballo
Tamaño

16G

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

master

489

0

STDIN

carlosf

r

12/29/2003

19:49:05

Cola1

MASTER

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.

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. There are quotas (limits on its utilization), so its use must be moderated.

Storage Systems

Every user has a SAHOME sub directory at their disposal from which they can access the massive storage system of CESGA. In this space they can introduce all kinds of files and data of less frequent usage they wish to keep. A backup of this system will be performed regularly. Files sent to this sub directory should not be sent to the queue. When a particular file is needed, it should be moved to the home directory.

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.

/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
Compilers and options:
1. Fortran compilers are f77, f90 and f95
2. C compiler is cc
3. C++ compiler is cxx
4. The recommended optimization option is -fast. It selects a set of options aimed at optimizing the code. As in every other optimization option, special attention should be paid to the results obtained with the code to check if they are correct before going into large computing.

Automatic parallelization with KAP compilers

The preprocessors of Kuck & Associates for Fortran and C are used to obtain the OpenMP directives that can be inserted into a sequential code to enable the SMP parallelism.
These optimizers use a Fortran 90, Fortran 77 or C code and they parallelize it automatically in those places where it is safe and possible. To use the compilers and parallelize, use the following calls:

```
kf90 -fkapargs=-conc prog.f
or
kcc -ckapargs=-conc prog.c
```

In the files prog.cmp.f, prog.cmp.c and prog.out will appear the modifications that have been introduced in the preprocessor in the original code. You can find more information about the different options of the compilers in the kf90 and kcc man pages.
To execute the code, it is necessary to set the environment variable OMP_NUM_THREADS to the number of processors that are going to be used (from 1 to 4). For example, to use 4 cpus:

```
export OMP_NUM_THREADS=4
```

Then we can run our code as we'll do it normally.

For more information about the Kap compilers, and their optimization and parallelization options check the Compaq documentation pages.

Mathematical Libraries
Compaq Extended Math Library (CXML)
It provides a set of mathematical subroutines specially optimized for the Alpha platform. It includes subroutines in the following areas:
Basic linear Algebra(BLAS): includes BLAS1, BLAS1E, BLAS1S, BLAS2 y BLAS3
Systems of linear equations and eigenvalues (LAPACK)
Systems of dispersed linear equations: include direct and interactive methods.
Signal processing: include FFTs, sin/cos transforms, convolutions, correlations, and digital filters.

The majority of routines include versions for real and complex numbers, as well as supporting simple and double precision. CXML also provide SCIPORT, an implementation of Compaq Computer Corporation of the CRAY SCILIB. This library provides 64 bit routines with simple precision to move applications from CRAY systems to Alpha systems.

The CXML library also includes a parallelized version (cxmlp) with the shared memory model of all subroutines (the parallelized version includes all subroutines, although some of them are not parallelized, the complete list of subroutines which are parallelized can be found here) To compile and link a program which contains call to the CXML routines, you must use:

f77 mi_programa.f -lcxml o cc mi_programa.c -lcxml

To use the parallelized version of the library:
f77 mi_programa.f -lcxmlp

To know how to run a program linked to the parallel version of the library, see the “running OpenMP jobs” section.

The complete documentation of the CXML library can be found aquí.

Compaq Portable Math Library (CPML)
Includes a wide range of mathematical functions: trigonometric functions, exponential functions, logarithms, algebraic functions, complex functions, etc... Click aquí to obtain more information.

Using MPI
MPI is a parallel programming interface for sending messages among parallel processes (for which it is necessary to have added the MPI code to the program). In order to enable MPI in the programs, it is necessary to include the MPI head file in the source code and to link to the MPI libraries.

Compilation and linking

For the Fortran codes, it is necessary to include the following directive to the the source code in every piece of code that uses MPI:

INCLUDE 'mpif.h'
and compile with the following command:
f90 miprograma.f -o miprograma.exe -lmpi

For C codes, it is necessary to add the following line:
#include <mpi.h>
and compile with the following command:
cc miprograma.c -o miprograma.exe -lmpi

Running MPI codes

It is necessary to use the command dmpirun to run every MPI code in interactive mode or in the batch queue system. For more information on how to run MPI codes from the batch queue system, check the batch queue system manual.

Using OpenMP
OpenMP is a set of extensions to the Fortran, C and C++ standards to support the parallel execution in the shared memory model. To use it, it is necessary to add directives to the source code to parallelize the loops and specify certain properties of the variables (Kap compilers incorporate automatically these type of directives using the -conc option).

Compiling and linking

To compile Fortran code that includes OpenMP directives:
f90 -fast -omp miprograma.f -o miprograma.exe
To compile C code that include OpenMP directives:
cc -fast -omp miprograma.c -o miprograma.exe

Running OpenMP jobs

To run OpenMP codes it is necessary to send the job to the queue. The environment variable, OMP_NUM_THREADS, is thus set to to the requested number of CPUs. For more information check the batch queue system use manual.

Common problems with OpenMP

One of the most common problems after parallelizing a code with OpenMP is the generation of floating point exceptions or segment violations that were not present before parallelization. This problem is often produced when the variables used are not properly initialized.