CNAG User's Guide

Barcelona Supercomputing Center

July 29, 2019

Contents

1 Introduction

This user guide for the CNAG cluster is intended to provide the minimum amount of information needed by a new user of this system. As such, it assumes that the user is familiar with many of the standard aspects of supercomputing such as the Unix operating system.

Here you can find most of the information you need to use our computing resources and the technical documentation about the machine. Please read carefully this document and if any doubt arises do not hesitate to contact us (Getting help (chapter 7)).

2 System Overview

The CNAG cluster is comprised of:

- 20 GenB nodes
  - 2 x Xeon E5–2670 (8 cores) @ 2.60 GHz
  - 8 x 16 GB DDR3 1600MHz (128 GB)
  - 394 GB local disk storage
- 108 GenC nodes
  - 2 x Xeon E5–2680v3 (12 cores) @ 2.50 GHz
  - 8 x 32 GB DDR4 2133 MHz (256 GB)
  - 420 GB local disk storage
- 12 GPUA nodes
  - 2 x Xeon E5–2640v3 (8 cores) @ 2.60 GHz
  - 2 x NVIDIA Tesla K80 GPU
  - 8 x 16 GB DDR4 2133 MHz (128 GB)
  - 867 GB local disk storage
- 1 SMP node
  - 4 x Xeon E7–4830v3 (12 cores) @ 2.1GHz
  - 64 x 32 GB DDR4 (2 TB)
  - 487 GB local disk storage

In addition to that, there are 2 Lustre High Performance Distributed Filesystems:

- /project (2 PB)
- /scratch (1.4 PB)

Nodes and storage are connected by Infiniband (High Speed Interconnect) for computation (MPI) and storage access. The Operating System in the cluster is Red Hat 6.7.

3 Connecting to CNAG cluster

First thing you should know is your username and password. Once you have a login and its associated password you can get into the cluster through one of the following login nodes:

```
login1 (172.16.10.20)
login2 (172.16.10.21)
```

You must use Secure Shell (ssh) tools to login into or transfer files to the cluster. We do not accept incoming connections from protocols such as telnet, ftp, rlogin, rcp, or rsh commands. To get more information about the Secure Shell version supported and how to get ssh for your system (including Windows systems) see Appendices (chapter 8).

Here you have two examples of logging into the cluster from a UNIX environment:

```
> ssh -l username login1
username's password:
Last login: Thu May 18 08:07:04 2016 from XXXX
username@login1:~>
```

or alternatively

```
> ssh -l username 172.16.10.20
username's password:
Last login: Thu May 18 08:07:04 2016 from XXXX
username@login1:~>
```

These addresses are only accesible inside CRG's network, comprising CNAG's offices and CRG's VPN connections.

As can be seen, you will be prompted for your password before access is granted. If you are on a Windows system, you need to download and install a Secure Shell client to perform the connection to the machine (See Appendices (chapter 8) for more information).

Most of these applications are graphical and you will have to fill some information in some of the fields offered. In the field 'Host name' or 'Remote Host name' you will need to introduce the URL of the login node. After entering your username and password as requested by the Secure Shell client you should be logged in.

The first time that you connect to the cluster Secure Shell exchanges some information to establish the communication. Then it requires that you accept of the RSA key of the remote host (this is just an unique identifier for the server). You must answer 'yes' to accept the key and connect. If you select 'no', the connection will be refused and terminated.

If you cannot get access to the system after following this procedure, first consult Appendices (chapter 8) for extended information about Secure Shell and, if the problem persists, please contact us (see Getting Help (chapter 7)).

Once inside the machine you will be presented with a UNIX shell prompt and you will normally be in your home directory ($HOME). If you are new to UNIX, you will have to learn the basics.

You can jump from one login node to the other but it is not possible to jump to a computing node. Outgoing connections of any kind are only possible from the login nodes.

3.1 Transferring files

The easiest way to tranfer files to/from the cluster is with the use of scp and sftp commands. You should execute them from your local machine. Here there are some examples of each of these tools transferring files to the cluster:

```
localsystem$ scp localfile username@172.16.10.20:
username's password:
localsystem$ sftp username@172.16.10.20
username's password:
sftp> put localfile
```

These are the ways to retrieve files from CNAG cluster to your local machine:

```
localsystem$ scp username@172.16.10.20:remotefile localdir
username's password:
localsystem$ sftp username@172.16.10.20
username's password:
sftp> get remotefile
```

On a Windows system, most of the Secure Shell clients come with a tool to make secure copies or secure ftp's. There are several tools that accomplishes the requirements, please refer to the Appendices (chapter 8) for the most common ones and examples of use.

Data Transfer Machine

We provide special machines for file transfer (required for large amounts of data). These machines are dedicated to Data Transfer, each one with a 10Gbits/s connection, and are accessible through ssh with the same account credentials as the cluster. They are:

- dt1.cnag.eu (172.16.10.23)
- dt2.cnag.eu (172.16.10.24)

3.2 Graphical applications

The only way to use graphical applications on the cluster is tunneling all the graphical traffic of the application you opened to your local machine through the Secure Shell connection you established to connect to the cluster. You will need to have an Xserver running on your local machine to be able to show the graphical information.

Most of the UNIX flavors have an X server installed by default. In a Windows environment, you will probably need to download and install some type of X server emulator (see Appendices (chapter 8)).

The second step in order to be able to execute graphical applications is to enable in your Secure Shell connection the forwarding of the graphical information through the secure channel created. This is normally done adding the '-X' flag to your normal ssh command used to connect to the cluster.

Here you have an example:

```
localsystem$ ssh -X -l username 172.16.10.20
username's password:
Last login: Fri Apr 23 15:07:04 2010 from XXXX
username@cn501:~>
```

For Windows systems you will have to enable the 'X11 forwarding' in your SSH client. That option normally resides on the 'Tunneling' or 'Connection' menu of the client configuration window. (See Appendices (chapter 8) for further details).

4 File Systems

IMPORTANT: It is your responsibility as a user of the CNAG cluster to backup all your critical data. We only guarantee a daily backup of user data under */home* and of the *qseq* and *fastq* files from finished runs of the sequencers.

Each user has several areas of disk space to store files and some of them may have size or time limits. Please read carefully this section to know about the usage policy of each of these filesystems.

There are 3 different types of storage available inside a node:

- NFS: NFS is a distributed file system protocol which allows the user access to */home* and */apps* from all nodes.
- Lustre: Lustre is a parallel distributed file system which can be accessed from all the nodes.
- Local hard drive: Every node has an internal hard drive accessible only from the node itself.

4.1 NFS

/home and */apps* are mounted in all nodes through NFS from one of the servers.

/apps contains all the software installed in the cluster for all users by support team. If you need an application/library or different version which can not be found there you can ask us for its installation (see Getting Help (chapter 7)). You should indicate the name and version of the desired application/ library, as well as the web page where it can be downloaded (more information about the available software at Software Environment (chapter 6)).

/home contains the personal directory of the users and is the directory where you start when logging into the cluster. Every user has its own home directory to store code, small files or applications which only they will use. As this is **NOT** a parallel file system (only 1 server) it is not recommended to use it as the working directory or output directory of production runs. Using it that way is likely

to hurt your jobs' performance and may harm other user's experience while using the cluster. You should use your personal space in Lustre for anything other than small tests. An incremental backup of these directories is performed daily.

4.2 Local Hard Drive

Every node has a local hard drive that can be used as a local scratch space to store temporary files during the execution of a job. This space is mounted over */scratch_tmp* directory. The amount of space within the */scratch_tmp* filesystem is about 160G in the compute nodes and 375G in the HiMem. All data stored in these local hard drives is only accessible from the node itself. In order to use the local hard drive from your jobs or commands, you should refer to it through the environment variable $TMPDIR.

4.3 Lustre

Lustre is a parallel distributed file system which provides a high performance and scalable file system. There are two Lustre filesystems available in CNAG cluster from all nodes:

- /project: This space is intended to store data that needs to be shared between the users of the same group or project.
- /scratch: This space is intended to store the temporary files of your jobs during its execution.

/project has a directory for each group of users while */scratch* has a directory per user. A quota per user is active in both Lustre file systems. You can check the limits of your quota with the command:

```
lfs quota -h <filesystem>
```

You should foresee the quantity of space that your jobs will generate and must make certain you have enough quota available to meet your jobs' needs. You can reduce your quota by archiving and compressing your data and by deleting old, intermediate or no longer useful data.

Whenever you are using the following commands anywhere in the cluster you should add these options to use the node's hard drive for temporary files:

```
sort -T $TMPDIR
java -Djava.io.tmpdir=$TMPDIR
```

Tips & tricks

It is a good practice to keep the filesystems as clean as possible. We recommend you remove any temporary files created during a jobs' execution at the end of the job or, if that's not possible, to store temporary data on specific paths so it's easy to find and delete afterwards. If the temporary data does not need to be available after the jobs' end you may consider using the nodes' local hard drive to store it (it doesn't count towards your quota and will be deleted automatically after the job ends).

If you try to write data into Lustre when you are near your quota the system slows down for all users that access the filesystem. Use the lquota command to monitor your quota usage before submitting the executions to avoid this situation.

If a directory holds a great number of files (more than 1024) its access will be slowed down. This can severely impact performance in a sudden way. It is preferrable to store as few files as possible or to distribute them among several subdirectories before assuming the performance penalty.

Another performance problem is caused by reading and writing many files at the same time in the same directory, e.g., when several similar jobs start executing at the same time or temporary intermediate files are created and destroyed in quick succession (*sort* command). In that case it's preferred that you create a per-job directory to store the output files, avoiding also the huge number of files in a directory at the same time, or to use the nodes' local hard drive.

Lustre FileSystem commands

The usual filesystem utilities don't interact well on Lustre filesystems and may cause some performance problems. Lustre provides its own version of some of those tools to efficiently access the filesystem without hurting performance. Whenever a wide filesystem operation needs to be carried out you should use these commands instead of their common counterparts. Check the commands' documentation for usage and information.

```
lfs find
lfs ls
lfs cp
```

5 Running Jobs

SLURM is the utility used at CNAG Cluster for batch processing support, so all jobs must be run through it. This section provides information for getting started with job execution at CNAG Cluster.

There are six partitions in the cluster:

- main: GenC nodes batch execution. Default partition.
- genB: GenB nodes batch execution.
- gpu: GPUA nodes batch execution.
- interactive: GenB nodes available for executing interactive applications.
- smp: SMP node batch execution.

If nothing is specified, all jobs go to the main partition where they have the latest hardware available. Interactive access is guaranteed through mnsh command and provides a terminal in the interactive node for interactive executions (shells, testing programs, large compilations, etc.).

5.1 Classes

Classes specify the Quality of Service (QOS) that a job should receive. Classes have a limit on the time of the jobs that can be submitted to the class and an associated priority. Highest priority classes have shorter runtimes than lower priority classes.

The user's default class is normal and allows jobs to run up to 1 day. In addition, there's a highprio class that grants a greater priority for executions that need to jump ahead of the normal queue. This queue is available to all users but it's usage is only allowed with justification for a few jobs at a time and incorrect use of this class will be prosecuted. The lowprio class allows up to 7 days of execution, but it has the lowest priority on the cluster to avoid it from blocking normal executions.

The limits for these queues are the following:

Table 1: Joblimits by queue

Class	Maximum Wall Clock	Maximum Number of Jobs	Priority
lowprio	7 days	Unlimited	lowest
normal	1 day	Unlimited	middle
highprio	1 day	Unlimited	highest

5.2 Submitting jobs

A job is the execution unit for the SLURM job scheduler. We have created wrappers to make easier the adaptation to the batch system. A job is defined by a text file containing a set of directives describing the job, and the commands to execute.

SLURM wrapper commands

These are the basic directives to submit jobs:

```
mnsubmit <job_script>
```

submits a *job script* to the queue system (see below for job script directives).
You can also submit a job that depends on the previous execution of another job:

```
mnsubmit -dep afterok:<job_id>:<job_id> <job_script>
```

The <job_script> will be executed after the correct completion of the jobs with id <job_id>

```
mnq
```

shows all the jobs submitted.

```
mncancel <job_id>
```

remove the job from the queue system, canceling the execution of the processes, if they were already running.

```
mnhold <job_id>
```

Hold the job and prevent it from starting. mnhold accepts a list of jobs separated by blanks.

```
mnhold --release <job_id>
```

Release the job that was held.

```
mnsh
```

It allocates a shell with a 8 hour limit for testing and debug purposes.

```
mnsh -x
```

It allocates an interactive shell in the same console (without X forwarding).

```
sacct
```

Show basic information of your running or completed jobs after the 00:00 hours of the same day. Option "-S " allows to change this to the specified date. Also, a lot more of information can be obtained with "–long" option.

sacct also provides information about the memory used on your jobs. If you want to know the maximum amount of memory per process that any of your jobs used, the commands to run would be:

```
sacct -o jobid,maxrss
sacct -j 1520 -o maxrss
```

Sacct is a very powerful command with a lot of different options. Please refer to the man page for further information. (man sacct).

Job directives

A job must contain a series of directives to inform the batch system about the characteristics of the job. These directives appear as comments in the job script, with the following syntax:

```
# @ directive = value
```

Additionally, the job script may contain a set of commands to execute. If not, an external script must be provided with the 'executable' directive. Here you may find the most common directives:

```
# @ class = class_name
```

With the class you can specify the priority of your job as is explained at Section 5.1.

```
# @ partition = himem
```

The partition where the job is to be submitted. Let this field empty unless you need to use special partitions such as "himem" or "development".

```
# @ wall_clock_limit = HH:MM:SS
```

The limit of wall clock time. This is a mandatoryfield and you must set it to a value greater than the real execution time for your application and smaller than the time limits granted to the user. Notice that your job will be killed after the elapsed period

```
# @ initialdir = pathname
```

The working directory of your job (i.e. where the job will run). If not specified, it is the current working directory at the time the job was submitted.

```
# @ error = file
```

The name of the file to collect the stderr output of the job.

```
# @ output = file
```

The name of the file to collect the standard output (stdout) of the job.

```
# @ total_tasks = number
```

The number of processes to start. Optionally, you can specify how many threads each process would open with the directive:

```
# @ cpus_per_task = number
```

The number of cpus assigned to the job will be the total_tasks number * cpus_per_task number.

```
# @ tasks_per_node = number
```

The number of tasks assigned to a node.

```
# @ memory = n
```

The amount of memory per cpu requested in MB. If it is not set, it will take 5,8GB per cpu requested.

```
# @ requeue = 1
```

With this directive, slurm will requeue the job if it died due to a node fail.

```
# @ scratch = n
```

Slurm guarantees that when the job starts there are at least n MB free in the local scratch of the allocated nodes.

There are also a few SLURM environment variables you can use in your scripts:

Table 2: SLURM environment variables

Variable	Meaning
SLURM_JOBID	Specifies the job ID of the executing job
SLURM_NPROCS	Specifies the total number of processes in the job
SLURM_NNODES	Is the actual number of nodes assigned to run your job
SLURM_PROCID	Specifies the MPI rank (or relative process ID) for the current process. The range is from 0-(SLURM_NPROCS–1)
SLURM_NODEID	Specifies relative node ID of the current job. The range is from 0-(SLURM_NNODES–1)
SLURM_LOCALID	Specifies the node-local task ID for the process within a job

Examples

Example for a sequential job :

```
#!/bin/bash
# @ job_name = test_serial
# @ initialdir = .
# @ output = serial_%j.out
# @ error = serial_%j.err
# @ total_tasks = 1
# @ wall_clock_limit = 00:02:00
./serial_binary > serial.out
```

The job would be submitted using:

```
> mnsubmit ptest.cmd
```

Examples for a parallel job :

```
#!/bin/bash
# @ job_name = test_parallel
# @ initialdir = .
# @ output = mpi_%j.out
# @ error = mpi_%j.err
# @ total_tasks = 8
# @ wall_clock_limit = 00:02:00
srun ./parallel_binary > parallel.output
```

6 Software Environment

All software and numerical libraries available at the cluster can be found at /apps/. If you need something that is not there please contact us to get it installed (see Getting Help (chapter 7)).

6.1 C Compilers

In the cluster you can find these C/C++ compilers :
icc /icpc -> Intel C/C++ Compilers

```
% man icc
% man icpc
```

gcc /g++ -> GNU Compilers for C/C++

```
% man gcc
% man g++
```

All invocations of the C or C++ compilers follow these suffix conventions for input files:

```
.C, .cc, .cpp, or .cxx -> C++ source file.
.c -> C source file
.i -> preprocessed C source file
.so -> shared object file
.o -> object file for ld command
.s -> assembler source file
```

By default, the preprocessor is run on both C and C++ source files.
These are the default sizes of the standard C/C++ datatypes on the machine

Table 3: Default datatype sizes on the machine

Type	Length (bytes)
bool (c++ only)	1
char	1
wchar_t	4
short	2
int	4
long	8
float	4
double	8
long double	16

Distributed Memory Parallelism

To compile MPI programs it is recommended to use the following handy wrappers: mpicc, mpicxx for C and C++ source code. You need to choose the Parallel environment first: module load openmpi /module load impi /module load poe. These wrappers will include all the necessary libraries to build MPI applications without having to specify all the details by hand.

```
% mpicc a.c -o a.exe
% mpicxx a.C -o a.exe
```

Shared Memory Parallelism

OpenMP directives are fully supported by the Intel C and C++ compilers. To use it, the flag -qopenmp must be added to the compile line.

```
% icc -qopenmp -o exename filename.c
% icpc -qopenmp -o exename filename.C
```

You can also mix MPI + OPENMP code using -openmp with the mpi wrappers mentioned above.

Automatic Parallelization

The Intel C and C++ compilers are able to automatically parallelize simple loop constructs, using the option "-parallel" :

```
% icc -parallel a.c
```

6.2 FORTRAN Compilers

In the cluster you can find these compilers :
ifort -> Intel Fortran Compilers

```
% man ifort
```

gfortran -> GNU Compilers for FORTRAN

```
% man gfortran
```

By default, the compilers expect all FORTRAN source files to have the extension ".f", and all FORTRAN source files that require preprocessing to have the extension ".F". The same applies to FORTRAN 90 source files with extensions ".f90" and ".F90".

Distributed Memory Parallelism

In order to use MPI, again you can use the wrappers mpif77 or mpif90 depending on the source code type. You can always man mpif77 to see a detailed list of options to configure the wrappers, ie: change the default compiler.

```
% mpif77 a.f -o a.exe
```

Shared Memory Parallelism

OpenMP directives are fully supported by the Intel Fortran compiler when the option "-qopenmp" is set:

```
% ifort -qopenmp
```

Automatic Parallelization

The Intel Fortran compiler will attempt to automatically parallelize simple loop constructs using the option "-parallel":

```
% ifort -parallel
```

6.3 Modules Environment

The Environment Modules package (http://modules.sourceforge.net/) provides a dynamic modification of a user's environment via modulefiles. Each modulefile contains the information needed to configure the shell for an application or a compilation. Modules can be loaded and unloaded dynamically, in a clean fashion. All popular shells are supported, including bash, ksh, zsh, sh, csh, tcsh, as well as some scripting languages such as perl.

Installed software packages are divided into five categories:

- Environment: modulefiles dedicated to prepare the environment, for example, get all necessary variables to use openmpi to compile or run programs
- Tools: useful tools which can be used at any time (php, perl, ...)
- Applications: High Performance Computers programs (GROMACS, ...)
- Libraries: Those are tipycally loaded at a compilation time, they load into the environment the correct compiler and linker flags (FFTW, LAPACK, ...)
- Compilers: Compiler suites available for the system (intel, gcc, ...)

Modules tool usage

Modules can be invoked in two ways: by name alone or by name and version. Invoking them by name implies loading the default module version. This is usually the most recent version that has been tested to be stable (recommended) or the only version available.

```
% module load intel
```

Invoking by version loads the version specified of the application. As of this writing, the previous command and the following one load the same module.

```
% module load intel/2017.1
```

The most important commands for modules are these:

- *module list* shows all the loaded modules
- *module avail* shows all the modules the user is able to load
- *module purge* removes all the loaded modules
- *module load <modulename>* loads the necessary environment variables for the selected modulefile (PATH, MANPATH, LD_LIBRARY_PATH...)
- *module unload <modulename>* removes all environment changes made by module load command
- *module switch <oldmodule> <newmodule>* unloads the first module (oldmodule) and loads the second module (newmodule)

You can run "module help" any time to check the command's usage and options or check the module(1) manpage for further information.

6.4 Debuggers

GDB (GNU DEBUGGER)

- /usr/bin/gdb

DDD (/apps/DDD)

- /apps/DDD/3.3.12/bin/ddd

6.5 Software in CNAG cluster

All software and numerical libraries available at CNAG cluster can be found at /apps. If you need something that is not available there contact us to get it installed (see Getting Help (chapter 7)).

7 Getting help

BSC provides excellent consulting assistance for the cluster users. User support consultants are available during normal business hours, Monday to Friday, 09 a.m. to 18 p.m. (CEST time).

User questions and support are handled at: cnag_support@bsc.es

If you need assistance, please send an email stating the nature of the issue, the date and time the issue occurred, and any other relevant information such as output files or error messages.

Please contact BSC if you have any questions or comments regarding policies or procedures.

Our mailing address is:

```
Barcelona Supercomputing Center - Centro Nacional de Supercomputación
C/ Jordi Girona, 31, Edificio Capilla
08034 Barcelona
```

7.1 Frequently Asked Questions (FAQ)

You can check the answers to most common questions at BSC's Support Knowledge Center[1]. There you will find online and updated versions of our documentation, including this guide, and a listing with deeper answers to the most common questions we receive as well as advanced specific questions unfit for a general-purpose user guide.

8 Appendices

8.1 SSH

SSH is a program that enables secure logins over an insecure network. It encrypts all the data passing both ways, so that if it is intercepted it cannot be read. It also replaces the old an insecure tools like telnet, rlogin, rcp, ftp,etc. SSH is a client-server software. Both machines must have ssh installed for it to work.

We have already installed a ssh server in our machines. You must have installed an ssh client in your local machine. SSH is available without charge for almost all versions of UNIX (including Linux and MacOS X). For UNIX and derivatives, we recommend using the OpenSSH client, downloadable from http://www.openssh.org, and for Windows users we recommend using Putty, a free SSH client that can be downloaded from http://www.putty.nl. Otherwise, any client compatible with SSH version 2 can be used.

This section describes installing, configuring and using the client on Windows machines. No matter your client, you will need to specify the following information:

- Select SSH as default protocol
- Select port 22
- Specify the remote machine and username

For example with putty client:

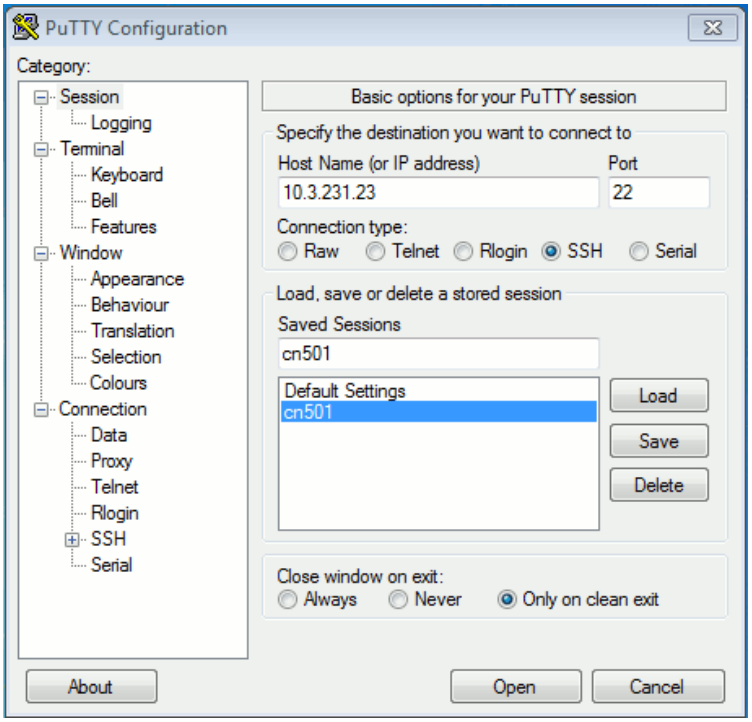

Figure 1: Putty client

This is the first window that you will see at putty startup. Once finished, press the **Open** button. If it is your first connection to the machine, your will get a *Warning* telling you that the host key from the server is unknown, and will ask you if you are agree to cache the new host key, press Yes.

[1] http://www.bsc.es/user-support/

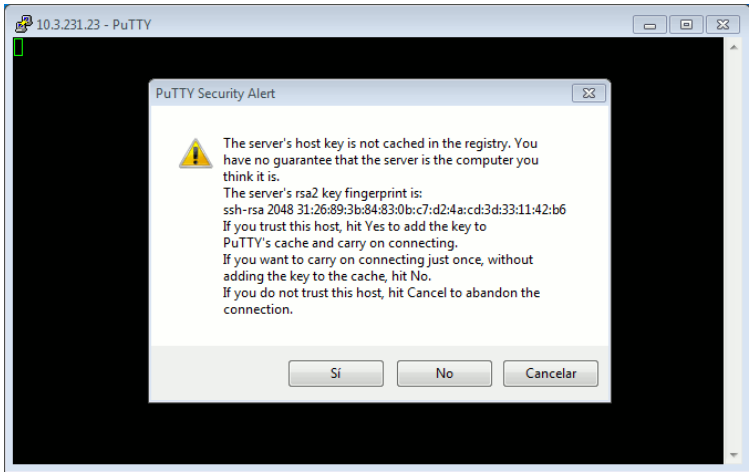

Figure 2: Putty certificate security alert

IMPORTANT: If you see this warning another time and you haven't modified or reinstalled the ssh client, please do *not* log in, and contact us as soon as possible (see Getting Help (chapter 7)).

Finally, a new window will appear asking for your login and password:

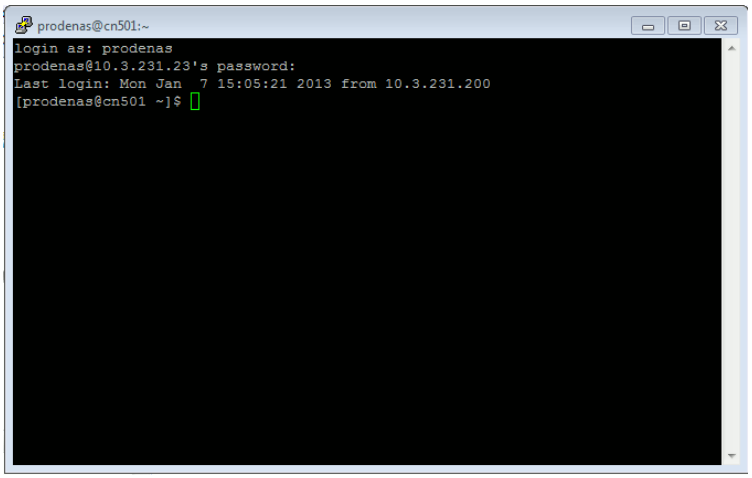

Figure 3: Cluster login

8.2 Transferring files

To transfer files to or from the cluster you need a secure ftp (sftp) o secure copy (scp) client. There are several different clients, but as previously mentioned, we recommend using of Putty clients for transferring files: **psftp** and **pscp**. You can find it at the same web page as Putty (http://www.putty.nl[2]).

Some other possible tools for users requiring graphical file transfers could be:

- WinSCP: Freeware Sftp and Scp client for Windows (http://www.winscp.net)
- SSH: Not free. (http://www.ssh.org)

Using WinSCP

WinSCP is the common software used at CNAG for file transferring via SFTP. The program first asks for the machine IP (172.16.10.20/16), and then for the username and password as you can see below

Once you are aunthenticated, you will be able to copy files from your local machine to CNAG, and from CNAG to your local machine. Simply you can drag'n'drop files from the left panel (your local machine) to the right panel (CNAG) and viceversa, as you can see below:

[2] http://www.putty.nl/

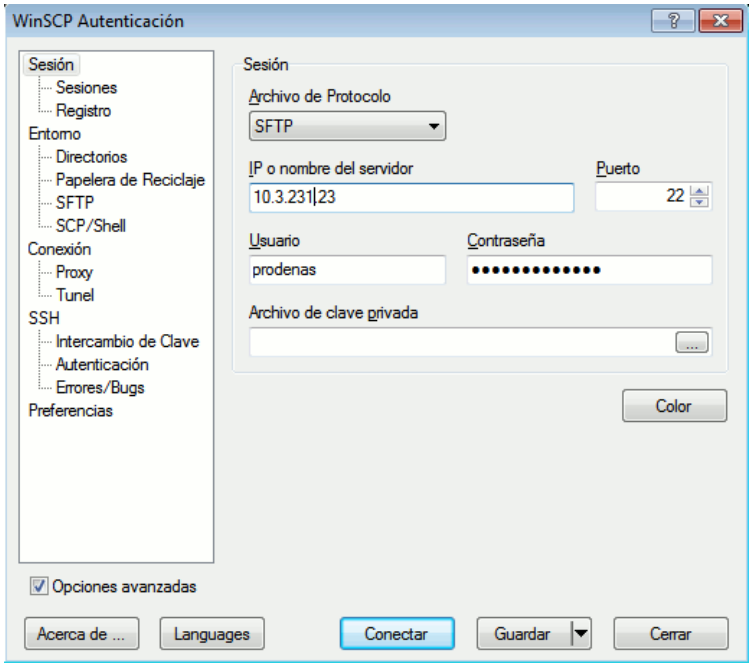

Figure 4: WinSCP authentication

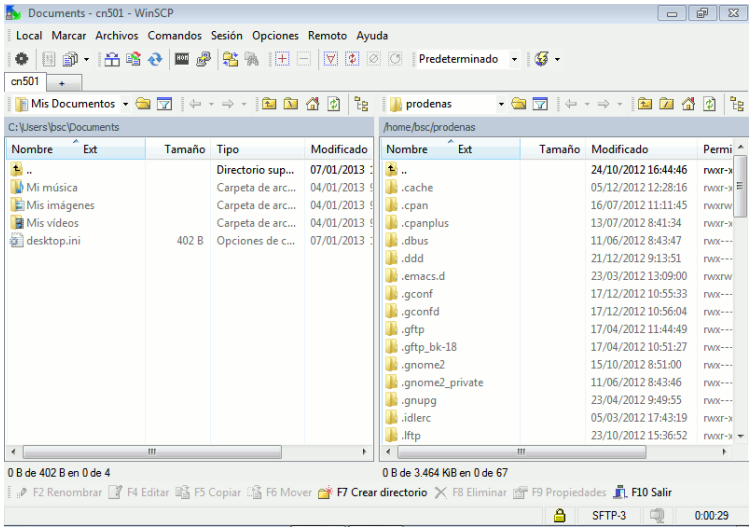

Figure 5: WinSCP transfer window

8.3 Using X11

In order to start remote X applications you need and X-Server running in your local machine. Here is a list of most common X-servers for windows:

- Cygwin/X: http://x.cygwin.com
- X-Win32 : http://www.starnet.com
- WinaXe : http://labf.com
- XconnectPro : http://www.labtam-inc.com
- Exceed : http://www.hummingbird.com

The only Open Source X-server listed here is Cygwin/X, you need to pay for the others. Once the X-Server is running run putty with X11 forwarding enabled:

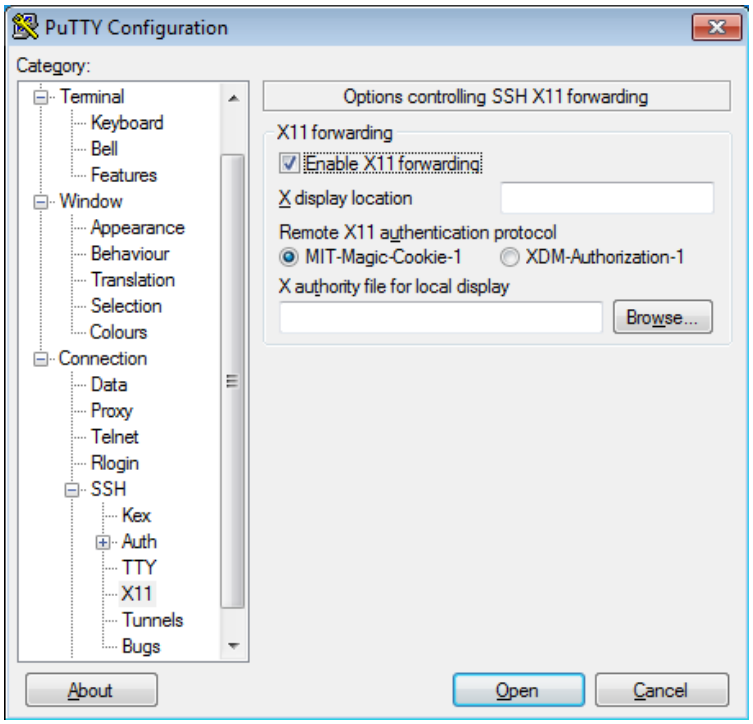

Figure 6: Putty X11 configuration