StarLife User's Guide

Barcelona Supercomputing Center

April 3, 2019

Contents

1 Introduction

This user's guide for the StarLife 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 features of supercomputing 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 6)).

2 System Overview

StarLife is a cluster based on Intel Xeon Gold processors from the Skylake generation with a Linux Operating System and an Infiniband interconnection network.

It has the following configuration:

- 2 login node and 36 compute nodes, each of them:
  - Dual socket Intel Xeon Gold 6138 CPU with 20 cores each @ 2.00GHz for a total of **40 cores per node**
  - 160GB of main memory **4 GB/core** (3 nodes fat memory, 8GB/core)
  - SSD 120GB as local storage
  - Dual 10 Gbit Ethernet
  - Intel Omni-Path network. Full fat tree topology.
  - GPFS via Intel Omni-Path

The operating system is SUSE Linux Enterprise Server 12 SP3.

The processors support well-known vectorization instructions such as SSE, AVX up to AVX–512[1].

2.1 Compilation for the architecture

To generate code that is optimized for the target architecture and the supported features such as SSE, MMX, AVX instruction sets you will have to use the corresponding compile flags. For compilations of MPI applications an MPI installation needs to be loaded in your session as well. For example Intel MPI via *module load impi*

Intel Compilers

The latest Intel compilers provide the best possible optimizations for the Xeon Gold architecture. By default, when starting a new session on the system the basic modules for the Intel suite will be automatically loaded. That is the compilers (intel/2018.3), the Intel MPI software stack (impi/2018.3) and the math kernel libraries MKL (mkl/2018.3) in their latest versions. We highly recommend linking against MKL where supported to achieve the best performance results.

To separately load the Intel compilers please use

```
module load intel
```

The corresponding optimization flags for icc are **CFLAGS="-xCORE-AVX512 -mtune=skylake"**. As the login nodes are of the exact same architecture as the compute node you can also use the flag **-xHost** which enables all possible optimizations available on the compile host.

GCC

The GCC provided by the system is version 4.8.5. For better support of new hardware features we recommend to use the latest version that can be loaded via the provided modules. Currently the latest version available in StarLife is GCC 8.1.0

```
module load gcc/8.1.0
```

The corresponding flags are **CFLAGS="-march=skylake-avx512"**

[1] https://software.intel.com/en-us/blogs/2013/avx-512-instructions

2.2 Login Nodes

You can connect to StarLife using two public login nodes. Please note that only incoming connections are allowed in the whole cluster. The logins are:

```
sl1.bsc.es
sl2.bsc.es
```

2.3 Password Management

In order to change the password, you have to login to a different machine (dt01.bsc.es). This connection must be established from your local machine.

```
% ssh -l username dt01.bsc.es

username@dtransfer1:~> passwd
Changing password for username.
Old Password:
New Password:
Reenter New Password:
Password changed.
```

Mind that that the password change takes about 10 minutes to be effective.

3 File Systems

IMPORTANT: It is your responsibility as a user of our facilities to backup all your critical data. *We only guarantee a daily backup of user data under /gpfs/home and a backup every two months for /gpfs/projects.*

Each user has several areas of disk space for storing files. These areas may have size or time limits, please read carefully all this section to know about the policy of usage of each of these filesystems. There are 3 different types of storage available inside a node:

- *Root filesystem*: Is the filesystem where the operating system resides
- *GPFS filesystems*: GPFS is a distributed networked filesystem which can be accessed from all the nodes.
- *Local hard drive*: Every node has an internal hard drive

3.1 GPFS Filesystem

The IBM General Parallel File System (GPFS) is a high-performance shared-disk file system providing fast, reliable data access from all nodes of the cluster to a global filesystem. GPFS allows parallel applications simultaneous access to a set of files (even a single file) from any node that has the GPFS file system mounted while providing a high level of control over all file system operations. In addition, GPFS can read or write large blocks of data in a single I/O operation, thereby minimizing overhead.

An incremental backup will be performed daily only for /gpfs/home and every two months for /gpfs/projects (not for /gpfs/scratch).

These are the GPFS filesystems available in the machine from all nodes:

- */apps*: Over this filesystem will reside the applications and libraries that have already been installed on the machine. Take a look at the directories to know the applications available for general use.
- */slgpfs/home*: This filesystem has the home directories of all the users, and when you log in you start in your home directory by default. Every user will have their own home directory to store own developed sources and their personal data. A default quota (section 3.3) will be enforced on all users to limit the amount of data stored there. Also, it is highly discouraged to run jobs from this filesystem. **Please run your jobs on your group's /gpfs/projects or /gpfs/scratch instead**.

- */slgpfs/projects*: In addition to the home directory, there is a directory in */gpfs/projects* for each group of users. For instance, the group bsc01 will have a */gpfs/projects/bsc01* directory ready to use. This space is intended to store data that needs to be shared between the users of the same group or project. A quota (section 3.3) per group will be enforced depending on the space assigned by Access Committee. It is the project's manager responsibility to determine and coordinate the better use of this space, and how it is distributed or shared between their users.

- */slgpfs/scratch*: Each user will have a directory over */gpfs/scratch*. Its intended use is to store temporary files of your jobs during their execution. A quota (section 3.3) per group will be enforced depending on the space assigned.

3.2 Local Hard Drive

Every node has a local solid state (SSD) that can be used as a local scratch space to store temporary files during executions of one of your jobs. This space is mounted over /scratch/tmp/$JOBID directory and pointed out by $TMPDIR environment variable. The amount of space within the /scratch filesystem is about 70 GB. All data stored in these local hard drives at the compute nodes will not be available from the login nodes. **You should use the directory referred to by $TMPDIR to save your temporary files during job executions. This directory will automatically be cleaned after the job finishes.**

3.3 Quotas

The quotas are the amount of storage available for a user or a groups' users. You can picture it as a small disk readily available to you. A default value is applied to all users and groups and cannot be outgrown.

You can inspect your quota anytime you want using the following command from inside each filesystem:

```
% bsc_quota
```

The command provides a readable output for the quota. Check BSC Commands (section 5.4) for more information.

If you need more disk space in this filesystem or in any other of the GPFS filesystems, the responsible for your project has to make a request for the extra space needed, specifying the requested space and the reasons why it is needed. For more information or requests you can Contact Us (chapter 6).

4 Running Jobs

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

4.1 Queues

There are several queues present in the machines and different users may access different queues. All queues have different limits in amount of cores for the jobs and duration. You can check anytime all queues you have access to and their limits using:

```
% bsc_queues
```

For longer and/or larger executions special queues are available upon request and will require proof of scalability and application performance. To solicit access to these special queues please contact us (chapter 6).

4.2 Submitting jobs

The method for submitting jobs is to use the SLURM *sbatch* directives directly.

A job is the execution unit for SLURM. A job is defined by a text file containing a set of directives describing the job's requirements, and the commands to execute.

In order to ensure the proper scheduling of jobs, there are execution limitations in the number of nodes and cores that can be used at the same time by a group. You may check those limits using command 'bsc_queues'. If you need to run an execution bigger than the limits already granted, you may [Contact Us]

SBATCH commands

These are the basic directives to submit jobs with *sbatch*:

```
sbatch <job_script>
```

submits a "job script" to the queue system (see Job directives (section 4.4)).

```
squeue
```

shows all the submitted jobs.

```
scancel <job_id>
```

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

4.3 Interactive Sessions

Allocation of an interactive session has to be done through SLURM:

```
salloc
```

4.4 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 and have to conform to either the *sbatch* syntaxes.

sbatch syxtax is of the form:

```
#SBATCH --directive=value
```

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

```
#SBATCH --qos=xlong
```

To request the queue for the job. If it is not specified, Slurm will use the user's default queue. The debug queue is only intended for small test.

```
#SBATCH --time=DD-HH:MM:SS
```

The limit of wall clock time. This is a mandatory field and you must set it to a value greater than 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 time has passed.

```
#SBATCH --workdir=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.

```
#SBATCH --error=file
```

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

```
#SBATCH --output=file
```

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

```
#SBATCH --nodes=number
```

The number of requested nodes.

```
#SBATCH --ntasks=number
```

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

```
#SBATCH --cpus-per-task=number
```

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

```
#SBATCH --tasks-per-node=number
```

The number of tasks assigned to a node.

```
#SBATCH --ntasks-per-socket=number
```

The number of tasks assigned to a socket.

```
#SBATCH --x11=batch
```

If it is set the job will be handled as graphical and Slurm will assign the necessary resources to the job, so you will be able to execute a graphical command and if you do not close the **current terminal** you will get a graphical window.

```
#SBATCH --reservation=reservation_name
```

The reservation where your jobs will be allocated (assuming that your account has access to that reservation). In some ocasions, node reservations can be granted for executions where only a se t of accounts can run jobs. Useful for courses.

```
#SBATCH --switches=number@timeout
```

By default, Slurm tries to schedule a job in order to use the minimum amount of switches. However, a user can request a maximum of switches for their jobs. Slurm will try to schedule the job for *timeout minutes*. If it is not possible to request number switches (each rack has 3 switches, every switch is connected to 24 nodes) after *timeout minutes*, Slurm will schedule the job by default.

```
#SBATCH --array=<indexes>
```

Submit a job array, multiple jobs to be executed with identical parameters. The indexes specification identifies what array index values should be used. Multiple values may be specified using a comma separated list and/or a range of values with a "-" separator. Job arrays will have two additional environment variable set. *SLURM_ARRAY_JOB_ID* will be set to the first job ID of the array. *SLURM_ARRAY_TASK_ID* will be set to the job array index value. For example:

```
sbatch --array=1-3 job.cmd
Submitted batch job 36
```

Will generate a job array containing three jobs and then the environment variables will be set as follows:

```
# Job 1
SLURM_JOB_ID=36
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=1

# Job 2
SLURM_JOB_ID=37
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=2

# Job 3
SLURM_JOB_ID=38
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=3
```

```
#SBATCH --exclusive
```

To request an exclusive use of a compute node without sharing the resources with other users. This only applies to jobs requesting less than one node (48 cores). All jobs with >= 48 cores will automatically use all requested nodes in exclusive mode.

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

For more information:

```
man sbatch
man srun
man salloc
```

4.5 Examples

sbatch examples

Example for a sequential job:

```
#!/bin/bash
#SBATCH --job-name="test_serial"
#SBATCH --workdir=.
#SBATCH --output=serial_%j.out
#SBATCH --error=serial_%j.err
#SBATCH --ntasks=1
#SBATCH --time=00:02:00
./serial_binary> serial.out
```

The job would be submitted using:

```
> sbatch ptest.cmd
```

Examples for a parallel job:

- Running a pure OpenMP job on one SL node using 40 cores on the xlong queue:

```
#!/bin/bash
#SBATCH --job-name=omp
#SBATCH --workdir=.
#SBATCH --output=omp_%j.out
#SBATCH --error=omp_%j.err
#SBATCH --cpus-per-task=40
#SBATCH --ntasks=1
#SBATCH --time=96:00:00
#SBATCH --qos=xlong
./openmp_binary
```

- Running on two SL nodes using a pure MPI job

```
#!/bin/bash
#SBATCH --job-name=mpi
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=80
srun ./mpi_binary
```

- Running a hybrid MPI+OpenMP job on two SL nodes with 20 MPI tasks (10 per node), each using 4 cores via OpenMP:

```
#!/bin/bash
#SBATCH --job-name=test_parallel
#SBATCH --workdir=.
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=20
#SBATCH --cpus-per-task=4
#SBATCH --tasks-per-node=10
#SBATCH --time=00:02:00
srun ./parallel_binary> parallel.output
```

Notifications

It is currently not possible to be notified about the status of jobs via email. To check if your jobs are being executed or have finished you will need to connect to the system and verify their status manually. For the future it is being planned to enable automatic notifications.

5 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 6)).

5.1 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 1: 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
```

5.2 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
```

5.3 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/2018.3
```

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.

5.4 BSC Commands

The Support team at BSC has provided some commands useful for user's awareness and ease of use in our HPC machines. A short summary of these commands follows:

- bsc_queues: Show the queues the user has access to and their time/resources limits.
- bsc_quota: Show a comprehensible quota usage summary for all accessible filesystems.

You can check more information about these commands through any of the following manpages:

```
% man bsc_commands
```

6 Getting help

BSC provides users with excellent consulting assistance. 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: support@bsc.es

If you need assistance, please supply us with the nature of the problem, the date and time that the problem occurred, and the location of any other relevant information, such as output files. Please contact BSC if you have any questions or comments regarding policies or procedures.

Our address is:

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

6.1 Frequently Asked Questions (FAQ)

You can check the answers to most common questions at BSC's Support Knowledge Center[2]. 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.

7 Appendices

7.1 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.org. 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:

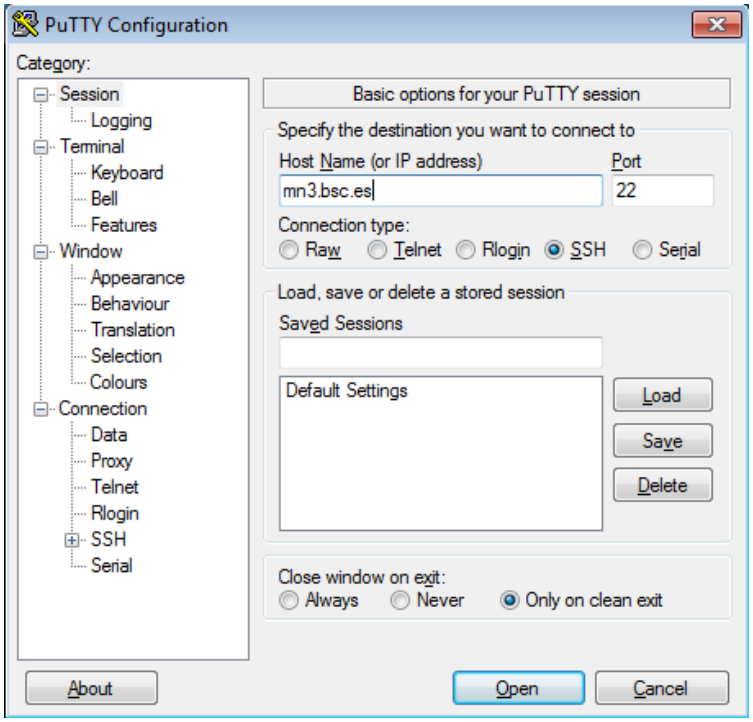

Figure 1: Putty client

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.

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

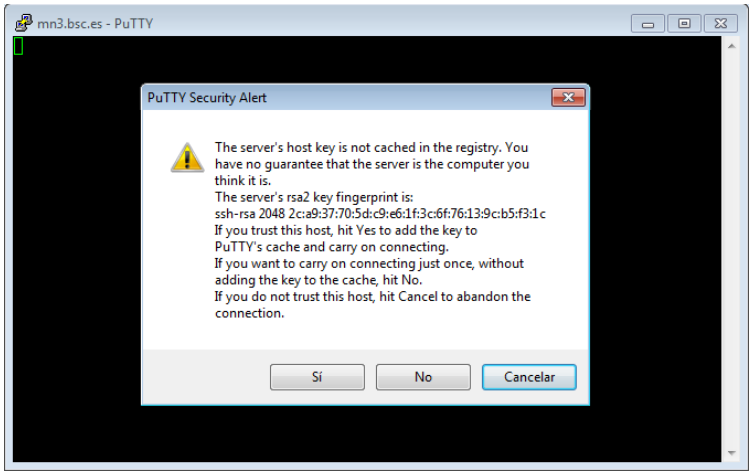

Figure 2: Putty certificate security alert

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 6)).

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

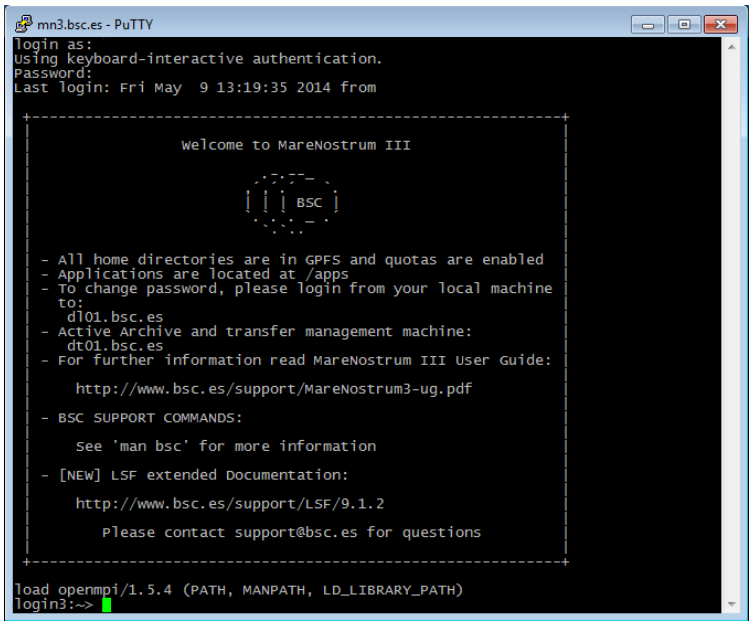

Figure 3: Cluster login

7.2 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.org[3]).

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)

[3] http://www.putty.org/

Using PSFTP

You will need a command window to execute psftp (press start button, click run and type cmd). The program first asks for the machine name (mn1.bsc.es), and then for the username and password. Once you are connected, it's like a Unix command line.

With command **help** you will obtain a list of all possible commands. But the most useful are:

- get file_name : To transfer from the cluster to your local machine.
- put file_name : To transfer a file from your local machine to the cluster.
- cd directory : To change remote working directory.
- dir : To list contents of a remote directory.
- lcd directory : To change local working directory.
- !dir : To list contents of a local directory.

You will be able to copy files from your local machine to the cluster, and from the cluster to your local machine. The syntax is the same that cp command except that for remote files you need to specify the remote machine:

```
Copy a file from the cluster:
> pscp.exe username@mn1.bsc.es:remote_file local_file
Copy a file to the cluster:
> pscp.exe local_file username@mn1.bsc.es:remote_file
```

7.3 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:

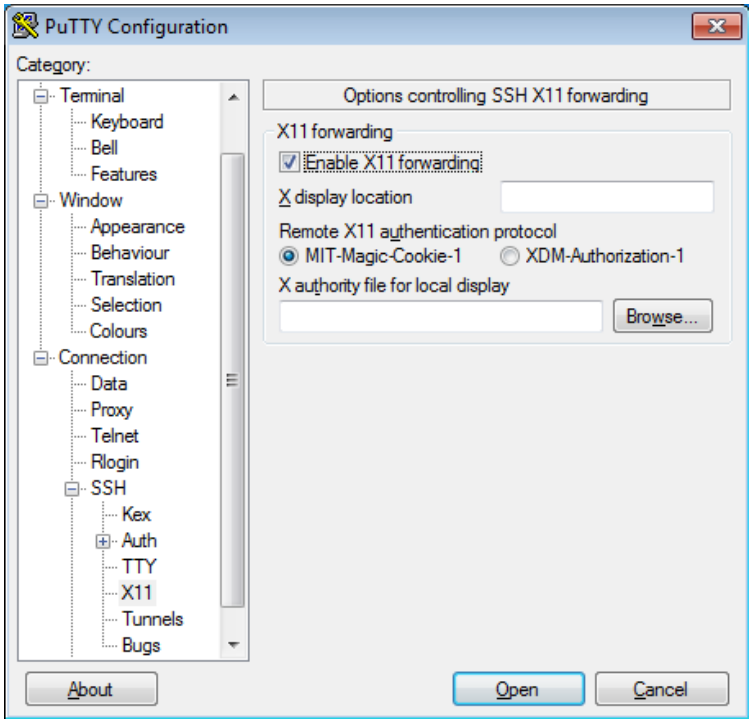

Figure 4: Putty X11 configuration