MinoTauro User's Guide

Barcelona Supercomputing Center

December 20, 2017

Contents

1 Introduction

This user's guide for the MinoTauro GPU 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 facilities, 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 8)).

2 System Overview

MinoTauro is a heterogeneous cluster with 2 configurations:

- 61 Bull B505 blades, each blade with the following technical characteristics:
  - 2 Intel E5649 (6-Core) processor at 2.53 GHz
  - 2 M2090 NVIDIA GPU Cards
  - 24 GB of Main memory
  - Peak Performance: 88.60 TFlops
  - 250 GB SSD (Solid State Disk) as local storage
  - 2 Infiniband QDR (40 Gbit each) to a non-blocking network
  - 14 links of 10 GbitEth to connect to BSC GPFS Storage
- 39 bullx R421-E4 servers, each server with:
  - 2 Intel Xeon E5–2630 v3 (Haswell) 8-core processors, (each core at 2.4 GHz,and with 20 MB L3 cache)
  - 2 K80 NVIDIA GPU Cards
  - 128 GB of Main memory, distributed in 8 DIMMs of 16 GB – DDR4 @ 2133 MHz - ECC SDRAM –
  - Peak Performance: 250.94 TFlops
  - 120 GB SSD (Solid State Disk) as local storage
  - 1 PCIe 3.0 x8 8GT/s, Mellanox ConnectX®–3FDR 56 Gbit
  - 4 Gigabit Ethernet ports.

The operating system is RedHat Linux 6.7 for both configurations.

3 Connecting to MinoTauro

The 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:

- mt1.bsc.es
- mt2.bsc.es

You must use Secure Shell (ssh) tools to login into or transfer files into the cluster. We do not accept incoming connections from protocols like telnet, ftp, rlogin, rcp, or rsh commands. Once you have logged into the cluster you cannot make outgoing connections for security reasons.

To get more information about the supported secure shell version and how to get ssh for your system (including windows systems) see Appendices (chapter 9).

Once connected to the machine, you will be presented with a UNIX shell prompt and you will normally be in your home ($HOME) directory. If you are new to UNIX, you will need to learn the basics before doing anything useful.

If you have used MinoTauro before, it's possible you will see a message similar to:

```
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
```

This message is displayed because the public SSH keys have changed, but the name of the logins has not. To solve this issue please execute (in your local machine):

```
ssh-keygen -f ~/.ssh/known_hosts -R 84.88.53.228
ssh-keygen -f ~/.ssh/known_hosts -R 84.88.53.229
ssh-keygen -f ~/.ssh/known_hosts -R mt1.bsc.es
ssh-keygen -f ~/.ssh/known_hosts -R mt1
ssh-keygen -f ~/.ssh/known_hosts -R mt2.bsc.es
ssh-keygen -f ~/.ssh/known_hosts -R mt2
```

3.1 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.

3.2 Transferring files

There are two ways to copy files from/to the Cluster:

- Direct scp or sftp to the login nodes
- Using a Data transfer Machine which shares all the GPFS filesystem for transferring large files

Direct copy to the login nodes.

As said before no connections are allowed from inside the cluster to the outside world, so all scp and sftp commands have to be executed from your local machines and never from the cluster. The usage examples are in the next section.

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 accomplish the requirements, please refer to the Appendices (chapter 9), where you will find 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 and are accessible through ssh with the same account credentials as the cluster. They are:

- dt01.bsc.es
- dt02.bsc.es

These machines share the GPFS filesystem with all other BSC HPC machines. Besides scp and sftp, they allow some other useful transfer protocols:

- scp

```
localsystem$ scp localfile username@dt01.bsc.es:
username's password:

localsystem$ sftp username@dt01.bsc.es
username's password:
sftp> put localfile
```

- sftp

```
localsystem$ scp username@dt01.bsc.es:remotefile localdir
username's password:
localsystem$ sftp username@dt01.bsc.es
username's password:
sftp> get remotefile
```

- BBCP

```
bbcp -V -z <USER>@dt01.bsc.es:<FILE> <DEST>
bbcp -V <ORIG>  <USER>@dt01.bsc.es:<DEST>
```

- FTPS

```
gftp-text ftps://<USER>@dt01.bsc.es
get <FILE>
put <FILE>
```

- GRIDFTP (only accessible from dt02.bsc.es)

Data Transfer on the PRACE Network

PRACE users can use the 10Gbps PRACE Network for moving large data among PRACE sites.
The selected data transfer tool is Globus/GridFTP[1] which is available on dt02.bsc.es
In order to use it, a PRACE user must get access to dt02.bsc.es:

```
% ssh -l pr1eXXXX dt02.bsc.es
```

Load the PRACE environment with 'module' tool:

```
% module load prace globus
```

Create a proxy certificate using 'grid-proxy-init':

```
% grid-proxy-init
Your identity: /DC=es/DC=irisgrid/O=bsc-cns/CN=john.foo
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until: Wed Aug  7 00:37:26 2013
pr1eXXXX@dtransfer2:~>
```

The command 'globus-url-copy' is now available for transferring large data.

```
globus-url-copy [-p <parallelism>] [-tcp-bs <size>] <sourceURL> <destURL>
```

Where:

[1] http://www.globus.org/toolkit/docs/latest-stable/gridftp/

- -p: specify the number of parallel data connections should be used (recommended value: 4)
- -tcp-bs: specify the size (in bytes) of the buffer to be used by the underlying ftp data channels (recommended value: 4MB)
- Common formats for sourceURL and destURL are:
  - file:///(on a local machine only) (e.g. file:///home/pr1eXX00/pr1eXXXX/myfile)
  - gsiftp://(e.g. gsiftp://supermuc.lrz.de/home/pr1dXXXX/mydir/)
  - remember that any url specifying a directory must end with /.

All the available PRACE GridFTP endpoints can be retrieved with the 'prace_service' script:

```
% prace_service -i -f bsc
gftp.prace.bsc.es:2811
```

More information is available at the PRACE website[2]

3.3 Active Archive Management

Active Archive (AA) is a mid-long term storage filesystem that provides 3.7 PB of total space. You can access AA from the Data Transfer Machine (section 3.2) (dt01.bsc.es and dt02.bsc.es) under /gpfs/archive/your_group.

NOTE: There is no backup of this filesystem. The user is responsible for adequately managing the data stored in it.

To move or copy from/to AA you have to use our special commands:

- dtcp, dtmv, dtrsync, dttar

These commands submit a job into a special class performing the selected command. Their syntax is the same than the shell command without 'dt' prefix (cp, mv, rsync, tar).

- dtq, dtcancel

dtq shows all the transfer jobs that belong to you. (works like mnq)
dtcancel works like mncancel (see below) for transfer jobs.

- *dttar:* submits a tar command to queues. Example: Taring data from /gpfs/to /gpfs/archive

```
% dttar -cvf  /gpfs/archive/usertest/outputs.tar ~/OUTPUTS
```

- *dtcp:* submits a cp command to queues. Remember to delete the data in the source filesystem once copied to AA to avoid duplicated data.

```
# Example: Copying data from /gpfs to /gpfs/archive
% dtcp -r  ~/OUTPUTS /gpfs/archive/usertest/
```

```
# Example: Copying data from /gpfs/archive to /gpfs
% dtcp -r  /gpfs/archive/usertest/OUTPUTS ~/
```

- *dtmv:* submits a mv command to queues.

```
# Example: Moving data from /gpfs to /gpfs/archive
% dtmv ~/OUTPUTS /gpfs/archive/usertest/
```

[2] http://www.prace-ri.eu/Data-Transfer-with-GridFTP-Details

```
# Example: Moving data from /gpfs/archive to /gpfs
% dtmv /gpfs/archive/usertest/OUTPUTS ~/
```

Additionally, these commands accept the following options:

- *–blocking:* Block any process from reading file at final destination until transfer completed.
- *–time:* Set up new maximum transfer time (Default is 18h).

It is important to note that these kind of jobs can be submitted from both the 'login' nodes (automatic file management within a production job) and 'dt01.bsc.es' machine. AA is only mounted in Data Transfer Machine (section 3.2). Therefore if you wish to navigate through AA directory tree you have to login into dt01.bsc.es

4 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 /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 and Data Transfer Machine (section 3.2)
- *Local hard drive*: Every node has an internal hard drive

4.1 Root Filesystem

The root file system is where the operating system is stored. It is NOT permitted the use of /tmp for temporary user data. The local hard drive can be used for this purpose Local Hard Drive (section 4.3).

4.2 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 /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.
- */gpfs/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 4.4) 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**.
- */gpfs/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 4.4) 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.

- */gpfs/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 4.4) per group will be enforced depending on the space assigned.

4.3 Local Hard Drive

Every node has a local SSD hard drive 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/directory and pointed out by $TMPDIR environment variable.

The amount of space within the /scratch filesystem varies depending on the configuration. The M2090 configuration (oldest one) has about 200 GB while the K80 configuration has about 100 GB.

All data stored in these local SSD hard drives at the compute nodes will not be available from the login nodes. Local hard drive data are not automatically removed, so each job has to remove its data before finishing.

4.4 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 commands from inside each filesystem:

```
% quota
% quota -g <GROUP>
% bsc_quota
```

The first command provides the quota for your user and the second one the quota for your group, showing the totals of both granted and used quota. The third command provides an easily readable summary for all filesystems.

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 8).

5 Graphical applications

You can execute graphical applications. To do that there are two ways depending on the purpose. You will need a X server 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 9))

5.1 Indirect mode (X11 forwarding)

This mode is intended for light graphical applications. It is made by tunneling all the graphical traffic through the established Secure Shell connection. It implies that no remote graphical resources are needed to draw the scene, the client desktop is responsible for that task. The way this communication is implemented implies the emulation of the X11 protocol, which implies a performance impact.

It is possible to execute graphical applications directly or using jobs. A job submitted in this way must be executed with the same ssh session.

In order to be able to execute graphical applications you have 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 mt1.bsc.es
username's password:
Last login: Fri Sep 2 15:07:04 2011 from XXXX
```

```
username@nvb127 ~]$ xclock
```

For Windows systems, you will have to enable the 'X11 forwarding' option that usually resides on the 'Tunneling' or 'Connection' menu of the client configuration window. (See Appendices (chapter 9) for further details).

5.2 Direct mode (VirtualGL)

This mode is suitable for any graphical applications, but the only way to use it is working with jobs because it is intended for applications which are GPU intensive.

VirtualGL wraps graphical applications and splits them into two tasks. Window rendering is done using X11 forwarding but OpenGL scenes are rendered remotely and then only the image is sent to the desktop client.

VirtualGL must be installed in the desktop client in order to connect the desktop with 'MinoTauro'. Please download the proper version here: http://www.bsc.es/support/VirtualGL/

Here you have an example:

```
localsystem$ vglconnect -s username@mt1.bsc.es
Making preliminary SSh connection to find a free
port on the server ...
username@mt1.bsc.es's password:
Making final SSh connection ...
username@mt1.bsc.es's password:
[VGL username@nvb127 ~]$ mnsubmit launch_xclock.sh
```

Jobs like 'launch_xclock.sh' must run the graphical application using the vglrun command:

```
vglrun -np 4 -q 70 xclock
```

It is also necessary to mark that job as 'graphical'. See Job Directives (section 6.2).

6 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.

6.1 Submitting jobs

There are 2 supported methods for submitting jobs. The first one is to use a wrapper maintained by the Operations Team at BSC that provides a standard syntax regardless of the underlying Batch system (*mnsubmit*). The other one is to use the SLURM *sbatch* directives directly. The second option is recommended for advanced users only.

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 cpus that cna 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 support@bsc.es.

Since MinoTauro is a cluster where more than 90% of the computing power comes from the GPUs, jobs that do not use them will have a lower priority than those that are gpu-ready.

SLURM wrapper commands

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

```
mnsubmit <job_script>
```

submits a "job script" to the queue system (see Job directives (section 6.2)).

```
mnq
```

shows all the submitted jobs.

```
mncancel <job_id>
```

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

```
mnsh
```

allocate an interactive session in the debug partition. You may add -c <ncpus> to allocate ncpu and/or -g to reserve a gpu. You may add -k80 so the interactive session has access to the new GPUS. By default, sessions are given on the old GPUs.

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 6.2)).

```
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.

Allocation of an interactive session in the debug partition has to be done through the mnsh wrapper:

```
mnsh
mnsh -k80
```

You may add -c <ncpus> to allocate ncpu and/or -g to reserve a gpu. You may add -k80 so the interactive session has access to the new GPUS. By default, sessions are given on the old GPUs.

6.2 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 *mnsubmit* or the *sbatch* syntaxes. Using *mnsubmit* syntax with *sbatch* or the other way around will result in failure.

mnsubmit syntax is of the form:

```
# @ directive = value
```

while *sbatch* 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:

```
# mnsubmit
# @ partition = debug
# @ class = debug
```

```
# sbatch
#SBATCH --partition=debug
#SBATCH --qos=debug
```

This partition is only intended for small tests.

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

```
# sbatch
#SBATCH --time=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.

```
# mnsubmit
# @ initialdir = pathname
```

```
# sbatch
#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.

```
# mnsubmit
# @ error = file
```

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

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

```
# mnsubmit
# @ output = file
```

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

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

```
# mnsubmit
# @ total_tasks = number
```

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

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

```
# mnsubmit
# @ cpus_per_task = number
```

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

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

```
# mnsubmit
# @ tasks_per_node = number
```

```
# sbatch
#SBATCH --ntasks-per-node=number
```

The number of tasks assigned to a node.

```
# mnsubmit
# @ gpus_per_node = number
```

```
# sbatch
#SBATCH --gres gpu:number
```

The number of GPU cards assigned to the job. This number can be [1–2] in *m2090* configurations or [1–4] on *k80* configurations. In order to allocate all the GPU cards in a node, you must allocate all the cores of the node. You must not request gpus if your job does not use them.

```
# mnsubmit
# @ features = <config>
```

```
# sbatch
#SBATCH --constraint=<config>
```

Select which configuration to run your job on. The valid values for *config* are either *m2090* or *k80*. If not specified, the job will run on the *m2090* configuration.

```
# mnsubmit
# @ X11 = 1
```

If it is set to 0, or it is not present, Slurm will handle that job as non-graphical. If it is set to 1 it will be handled as graphical and Slurm will assign the necessary resources to the job. There is no *sbatch* equivalent.

```
# mnsubmit
# @ switches = "number@timeout"
```

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

By default, Slurm schedules a job in order to use the minimum amount of switches. However, a user can request a specific network topology in order to run his job. Slurm will try to schedule the job for timeout minutes. If it is not possible to request number switches (from 1 to 14) after timeout minutes, Slurm will schedule the job by default.

```
# mnsubmit
# @ intel_opencl = 1
```

By default, Only Nvidia OpenCL driver is loaded to use the GPU device. In order to use the CPU device to run OpenCL, this directive must be added to the job script. As it introduces important changes in the Operating System setup, it is mandatory to allocate full nodes to use this feature. There are also a few SLURM environment variables you can use in your scripts:

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

mnsubmit 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= 12
# @ gpus_per_node= 2
# @ cpus_per_task= 1
# @ wall_clock_limit = 00:02:00
srun ./parallel_binary> parallel.output
```

Example for a job using the new K80 GPUs:

```
#!/bin/bash
# @ job_name= test_k80
# @ initialdir= .
# @ output= k80_%j.out
# @ error= k80_%j.err
# @ total_tasks= 16
# @ gpus_per_node= 4
# @ cpus_per_task= 1
# @ features = k80
# @ wall_clock_limit = 00:02:00
srun ./parallel_binary_k80> parallel.output
```

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:

```
#!/bin/bash
#SBATCH --job-name=test_parallel
#SBATCH --workdir=.
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=12
#SBATCH --gres gpu:2
#SBATCH --cpus-per-task=1
#SBATCH --time=00:02:00
srun ./parallel_binary> parallel.output
```

Example for a job using the new K80 GPUs:

```
#!/bin/bash
#SBATCH --job-name=test_k80
#SBATCH --workdir=.
#SBATCH --output= k80_%j.out
#SBATCH --error= k80_%j.err
#SBATCH --ntasks=16
#SBATCH --gres gpu:4
#SBATCH --cpus-per-task=1
#SBATCH --constraint=k80
#SBATCH --time=00:02:00
srun ./parallel_binary_k80> parallel.output
```

7 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 8)).

7.1 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 -openmp must be added to the compile line.

```
% icc -openmp -o exename filename.c
% icpc -openmp -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
```

7.2 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 "-openmp" is set:

```
% ifort -openmp
```

Automatic Parallelization

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

```
% ifort -parallel
```

7.3 Haswell compilation

To produce binaries optimized for the Haswell CPU architecture you should use either Intel compilers or GCC version 4.9.3 bundled with binutils version 2.26. You can load a GCC environment using module:

```
module load gcc/4.9.3
```

7.4 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/16.0.2
```

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.

7.5 TotalView

TotalView is a graphical portable powerful debugger from Rogue Wave Software designed for HPC environments. It also includes MemoryScape and ReverseEngine. It can debug one or many processes and/or threads. It is compatible with MPI, OpenMP, Intel Xeon Phi and CUDA.

Users can access to the latest version of TotalView 8.13 installed in:

```
/apps/TOTALVIEW/totalview
```

Important: Remember to access with ssh -X to the cluster and submit the jobs to x11 queue since TotalView uses a single window control.

There is a Quick View of TotalView[3] available for new users. Further documentation and tutorials can be found on their website[4] or in the cluster at:

```
/apps/TOTALVIEW/totalview/doc/pdf
```

7.6 Tracing jobs with BSC Tools

In this section you will find an introductory guide to get execution traces in Minotauro. The tracing tool Extrae supports many different tracing mechanisms, programming models and configurations. For detailed explanations and advanced options, please check the complete Extrae User Guide[5].

The most recent stable version of Extrae is always located at:

```
/apps/CEPBATOOLS/extrae/latest/default/64
```

[3] https://www.bsc.es/support/TotalView-QuickView.pdf
[4] http://www.roguewave.com/products/totalview.aspx
[5] http://www.bsc.es/computer-sciences/performance-tools/trace-generation/extrae/extrae-user-guide

This package is compatible with the default MPI runtime in MinoTauro (Bull MPI). Packages corresponding to older versions and enabling compatibility with other MPI runtimes (OpenMPI, MVAPICH) can be respectively found under this directory structure:

```
/apps/CEPBATOOLS/extrae/<choose-version>/<choose-runtime>/64
```

In order to trace an execution, you have to load the module extrae and write a script that sets the variables to configure the tracing tool. Let's call this script trace.sh. It must be executable (chmod +x ./trace.sh). Then your job needs to run this script before executing the application.

Example for MPI jobs:

```
#!/bin/bash
# @ output = tracing.out
# @ error = tracing.err
# @ total_tasks = 4
# @ cpus_per_task = 1
# @ tasks_per_node = 12
# @ wall_clock_limit = 00:10

module load extrae

srun ./trace.sh ./app.exe
```

Example for threaded (OpenMP or pthreads) jobs:

```
#!/bin/bash
# @ output = tracing.out
# @ error = tracing.err
# @ total_tasks = 1
# @ cpus_per_task = 1
# @ tasks_per_node = 12
# @ wall_clock_limit = 00:10

module load extrae

./trace.sh ./app.exe
```

Example of trace.sh script:

```
#!/bin/bash

export EXTRAE_CONFIG_FILE=./extrae.xml
export LD_PRELOAD=${EXTRAE_HOME}/lib/<tracing-library>
$*
```

Where:

- EXTRAE_CONFIG_FILE points to the Extrae configuration file. Editing this file you can control the type of information that is recorded during the execution and where the resulting trace file is written, among other parameters. By default, the resulting trace file will be written into the current working directory. Configuration examples can be found at:

  ${EXTRAE_HOME}/share/examples

- <tracing-library> depends on the programming model the application uses:

* Jobs that make explicit calls to the Extrae API do not load the tracing library via LD_PRELOAD, but link with the libraries instead.

** Jobs using automatic instrumentation via Dyninst neither load the tracing library via LD_PRELOAD nor link with it.

For other programming models and their combinations, check the full list of available tracing libraries at section 1.2.2 of the Extrae User Guide[6]

8 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).

[6] http://www.bsc.es/computer-sciences/performance-tools/trace-generation/extrae/extrae-user-guide

Job type	Tracing library	An example to get started
MPI	libmpitrace.so (C codes) libmpitracef.so (Fortran codes)	MPI/ld-preload/job.lsf
OpenMP	libomptrace.so	OMP/run_ldpreload.sh
Pthreads	libpttrace.so	PTHREAD/README
CUDA	libcudatrace.so	CUDA/ run_instrumented.sh
MPI+CUDA	libcudampitrace.so (C codes) libcudampitracef.so (Fortran codes)	MPI+CUDA/ld-preload/ job.slurm
OmpSs	-	OMPSS/job.lsf
Sequential job (manual instrumentation)	libseqtrace.so	SEQ/ run_instrumented.sh
*Automatic instrumentation of user functions and parallel runtime calls	-	SEQ/run_dyninst.sh

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
```

8.1 Frequently Asked Questions (FAQ)

You can check the answers to most common questions at BSC's Support Knowledge Center[7]. 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.

9 Appendices

9.1 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:

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

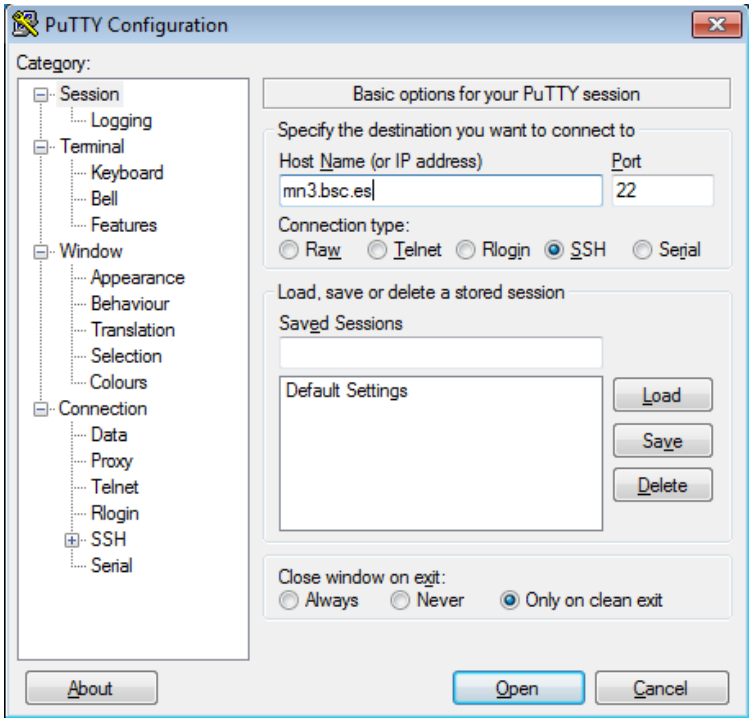

Figure 1: Putty client

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.

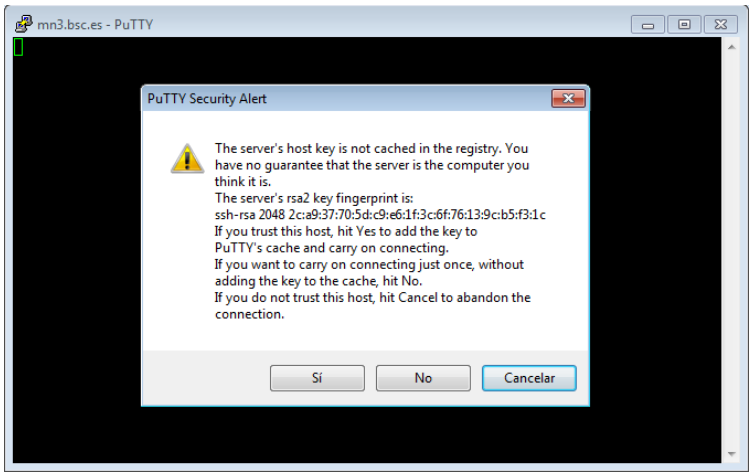

Figure 2: Putty certificate security alert

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 8)).

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

9.2 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[8]).

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

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

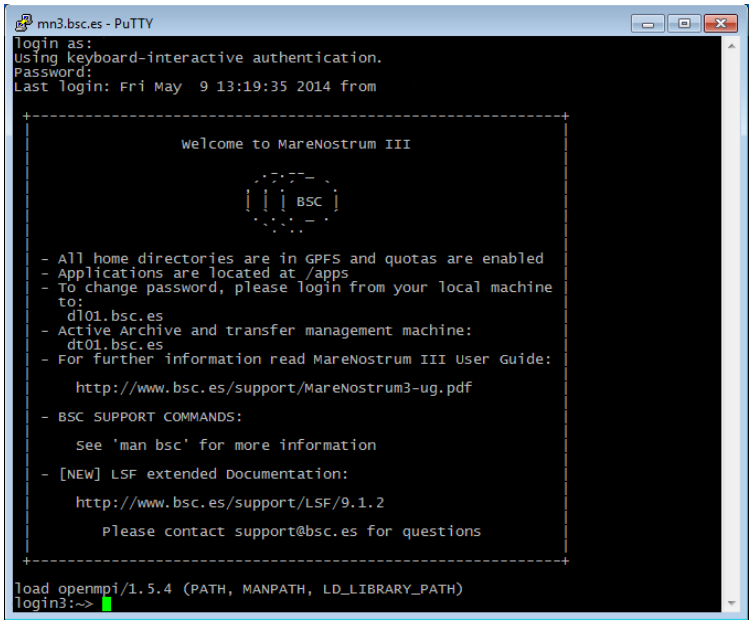

Figure 3: Cluster login

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

9.3 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:

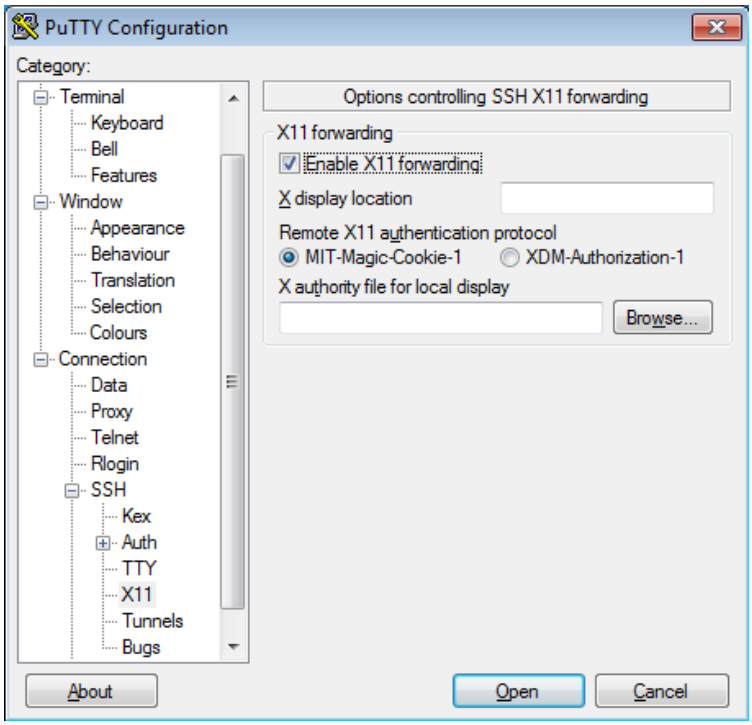

Figure 4: Putty X11 configuration