MareNostrum III User's Guide

Barcelona Supercomputing Center

April 28, 2015

Contents

1 Introduction

This user's guide for the MareNostrum III 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 7)).

2 System Overview

MareNostrum III is a supercomputer based on Intel SandyBridge processors, iDataPlex Compute Racks, a Linux Operating System and an Infiniband interconnection.

The current Peak Performance is 1.1 Petaflops. The total number of processors is 48,896 Intel SandyBridge-EP E5–2670 cores at 2.6 GHz (3,056 compute nodes) with 103.5 TB of main memory. See below a summary of the system:

- 37 iDataPlex compute racks. Each one composed of:
  - 84 IBM dx360 M4 compute nodes
  - 4 Mellanox 36-port Managed FDR10 IB Switches
  - 2 BNT RackSwitch G8052F (Management Network)
  - 2 BNT RackSwitch G8052F (GPFS Network)
  - 4 Power Distribution Units
- All IBM dx360 M4 node contain:
  - 2x E5–2670 SandyBridge-EP 2.6GHz cache 20MB 8-core
  - 500GB 7200 rpm SATA II local HDD
- Nodes are differentiated as follows:
  - RAM:
    - 64 nodes contain 8x 16G DDR3–1600 DIMMS (8GB/core) Total: 128GB/node
    - 64 nodes contain 16x 4G DDR3–1600 DIMMs (4GB/core) Total: 64GB/node
    - 2880 nodes contain 8x 4G DDR3–1600 DIMMs (2GB/core) Total: 32GB/node
  - 42 heterogenenous nodes contain:
    - 8x 8G DDR3–1600 DIMMs (4GB/core) Total: 64GB/node
    - 2x Xeon Phi 5110P accelerators
- 1.9 PB of GPFS disk storage
- Interconnection Networks
  - Infiniband Mellanox FDR10: High bandwidth network used by parallel applications communications (MPI)
  - Gigabit Ethernet: 10GbitEthernet network used by the GPFS Filesystem.
- Operating System: Linux - SuSe Distribution 11 SP3

3 Connecting to MareNostrum III

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:

- mn1.bsc.es

- mn2.bsc.es
- mn3.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 the Appendices (chapter 8). 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.

3.1 Password Management

In order to change the password, you have to login to a different machine (dl01.bsc.es). This connection must be established from your local machine.

```
% ssh -l username dl01.bsc.es

username@dlogin1:~> 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.

Here there are some examples of each of this tools transferring small files to the cluster:

```
localsystem$ scp localfile username@mn1.bsc.es:
username's password:

localsystem$ sftp username@mn1.bsc.es
username's password:
sftp> put localfile
```

These are the ways to retrieve files from the login nodes to your local machine:

```
localsystem$ scp username@mn1.bsc.es:remotefile localdir
username's password:
localsystem$ sftp username@mn1.bsc.es
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 accomplish the requirements, please refer to the Appendix A, 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 Transfers 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:

- 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:

- -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:

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

- 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 more than 3 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/
```

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

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

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, where the operating system is stored doesn't reside in the node, this is a NFS filesystem mounted from one of the servers.

As this is a remote filesystem only data from the operating system has to reside in this filesystem. It is NOT permitted the use of /tmp for temporary user data. The local hard drive can be used for this purpose as you could read in 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 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/tmp directory and pointed out by $TMPDIR environment variable. The amount of space within the /scratch filesystem is about 500 GB. All data stored in these local 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 a more readable output for the quota. Check BSC Commands (section 6.4) 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 7).

5 Running Jobs

LSF[3] is the utility used at MareNostrum III for batch processing support, so all jobs must be run through it. This document provides information for getting started with job execution at the Cluster.

5.1 LSF Commands

These are the basic commands to submit, control and check your jobs:

```
bsub < job_script
```

submits a "job script" passed through standard input (STDIN) to the queue system. Job directives (section 5.2) explains the available options to write a jobscript

```
bjobs [-w][-X][-l job_id]
```

shows all the submitted jobs.

```
bkill <job_id>
```

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

```
bsc_jobs
```

shows all the pending or running jobs from your group. Check BSC Commands (section 6.4) for more information.

[3] http://www.bsc.es/support/LSF/9.1.2

5.2 Job directives

A job must contain a series of directives to inform the batch system about the characteristics of the job. We encourage that you read the bsub command's manual from any of MareNostrum's terminals:

```
% man bsub
```

Here we provide a short summary of most common directives:

```
#BSUB -J job_name
```

Specify the name (description) of the job.

```
#BSUB -q debug
```

Specify the queue (section 5.5) for the job to be submitted. The *debug* queue is only intended for small tests, so there is a limit of 1 job per user, using up to 64 cpus (4 nodes), and one hour of wall clock limit. The queue might be reassigned by LSF internal policy, as with the *sequential (section 5.5)* queue.

```
#BSUB -W HH:MM
```

Specify how much time the job will be allowed to run. This is a **mandatory** field. NOTE: take into account that you can not specify the amount of seconds in LSF. 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

```
#BSUB -cwd 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.

```
#BSUB -e/-eo file
```

The name of the file to collect the stderr output of the job. You can use %J for job_id. -e option will APPEND the file, -eo will REPLACE the file.

```
#BSUB -o/-oo file
```

The name of the file to collect the standard output (stdout) of the job. -o option will APPEND the file, -oo will REPLACE the file.

```
#BSUB -n number
```

The number of tasks for the job. In MPI executions corresponds to the number of MPI processes and for sequential executions the number of cores.

```
#BSUB -x
```

Use the nodes exclusively. This is the default behaviour except for sequential (section 5.5) executions.

```
#BSUB -M number
```

Specify maximum amount of memory necessary for each task, specified in MB. This option is used to better schedule the jobs and select the compute nodes adequate to fulfill the job's needs. By default 1800 MB are reserved per task. Exclusive jobs will still get all available memory. You may check this FAQ[4] for a more in-depth explanation. **Note: For non-LowMem requests you must specify a ptile of 16.**

[4] http://www.bsc.es/user-support/faq.php#therearesomespecialrequirementsforjobs.howshouldiputthem

```
#BSUB -R "span[ptile=number]"
```

The number of processes assigned to a node. Note that full nodes will be allocated except for sequential (section 5.5) executions. Examples:

```
# if you want to use 4 processes per node and 4 threads:

#BSUB -R "span[ptile=4]"
export OMP_NUM_THREADS=4
```

```
# if your program has high memory
# consumption you can reduce the number
# of processes per node

#BSUB -R "span[ptile=14]"
```

5.3 MPI particulars

There are different MPI implementations available for usage. Some of them present special requirements to be used.

OpenMPI

OpenMPI is an Open Source MPI implementation and is loaded by default on our machines. You can use it like this:

```
% module load openmpi
% mpicc / mpif90 your_files
```

Once compiled, your jobscript must invoke mpirun:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J openmpi_example
#BSUB -W 00:05

module load openmpi
mpirun binary.exe
```

IntelMPI

IntelMPI is an MPI implementation developed by Intel. You can use it like this:

```
% module load intel impi
% mpicc / mpif90 your_files
```

Once compiled, your jobscript must invoke mpirun:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J impi_example
#BSUB -W 00:05

module load impi
mpirun binary.exe
```

IBM POE

IBM POE is an alternative MPI library binary compatible with IntelMPI. Use POE for executions with a large number of cores. First, in order to compile your MPI code using IBM POE:

```
% module load poe
% mpicc / mpif90 your_files
```

Once compiled, your jobscript must mention it explicitly:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J poe_example
#BSUB -W 00:05
#BSUB -a poe

module load poe
poe binary.exe
```

5.4 Jobscript examples

Purely sequential

```
#!/bin/bash

#BSUB -n 1
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J sequential
#BSUB -W 00:05

./serial.exe
```

Sequential with OpenMP threads

```
#!/bin/bash

#BSUB -n 16
#BSUB -R "span[ptile=16]"
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J sequential_OpenMP
#BSUB -x
#BSUB -W 00:05

export OMP_NUM_THREADS=16
./serial.exe
```

Parallel using MPI

```
#!/bin/bash

#BSUB -n 128
#BSUB -o output_%J.out
#BSUB -e output_%J.err
# In order to launch 128 processes with
# 16 processes per node:
#BSUB -R "span[ptile=16]"
#BSUB -J WRF.128-4
#BSUB -W 02:00

# You can choose the parallel environment through modules
module load intel openmpi
```

```
mpirun ./wrf.exe
```

Parallel using MPI and OpenMP threads

```
#!/bin/bash

# The total number of processes:
# 128 MPI processes + 2 OpenMP threads/process = 256 cores
#BSUB -n 128            # processes
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err


  ##########################################################

# This will allocate 8 processes per node so we have #
# 8 cores per node for the threads                   #

  ##########################################################

#BSUB -R "span[ptile=8]"

# exclusive mode (enabled by default)
#BSUB -x

#########################################################
# Then (128 MPI tasks) / (8 tasks/node) = 16 nodes  #
# reserved                                          #
# 16 nodes * 16 cores/node = 256 cores allocated    #
# (Matches the amount of cores asked for above)     #
#########################################################

#BSUB -J WRF.128-4
#BSUB -W 02:00

# Clean your environment modules
module purge

# You can choose the parallel environment through
# modules
module load intel openmpi

# 8 MPI processes per node and 16 cpus available
# (2 threads per MPI process):
export OMP_NUM_THREADS=2

mpirun ./wrf.exe
```

Parallel using MPI and extra memory

The following example shows how to ask for 3000 MB/task instead of the default 1800 MB/task. As no '-R' options are included it will still put 16 processes on each node, so the Common (2 GB/core) nodes won't be able to execute this job. This effectively guarantees that either MedMem (64 nodes with 4 GB/core) or HighMem (64 nodes with 8 GB/core) nodes will be used, or a mixture of both.

```
#!/bin/bash

# Total number of tasks
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err

# Requesting 3000 MB per task
# As no ptile is specified, only Medium and High memory nodes
# are elegible for this execution (max 128 nodes, 2048 cores)
#BSUB -M 3000

#BSUB -W 01:00
```

```
module purge
module load intel openmpi

mpirun ./my_mpi.exe
```

The following example requests the same memory per task as the preceding one but it is set to be executed on any nodes. This is done by reducing the amount of tasks per node so the total memory requested by all the tasks in the same node is below the LowMem memory threshold. This would be necessary to execute jobs that need more nodes or cpus than the MedMem and HighMem nodes can provide.

```
#!/bin/bash

# Total number of tasks
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err

# Requesting 3000 MB per task
#BSUB -M 3000

# Only 9 task per node.
# All compute nodes are available for execution
#BSUB -R "span[ptile=9]"

#BSUB -W 01:00

module purge
module load intel openmpi

mpirun ./my_mpi.exe
```

5.5 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
```

Check BSC Commands (section 6.4) for more information.

Sequential executions

For any job that requires a node or less resources, the sequential queue is automatically applied. This queue is the only one that uses the same node for more than one job at a time. It also has the least priority in the machine and the number of concurrently executed sequential jobs is limited to avoid disturbing large jobs' execution.

If you request 16 processes per node or use -x option, the full node will be assigned to your job. If you have memory problems, mind to specify the exclusive flag because just specifying a ptile may still share the node with other users.

5.6 MareNostrumHybrid (Intel Xeon Phi)

MareNostrum III has Intel Xeon Phi accelerators accessible for our users. If you have been granted access to them, you should be able to see the queues 'mic' and 'mic-interactive' in the output of bsc_queues. Check BSC Commands (section 6.4) for more information.

There are 42 nodes with Xeon Phi, 6 nodes available for interactive usage and 36 nodes for off-line executions. These nodes have:

- 2x Intel SandyBridge-EP E5–2670/1600 20M 8-core at 2.6 GHz
- 2x Xeon Phi 5110 P

- 8x8GB DDR3–1600 DIMM (4GB/core)

Right now the communication network between the nodes is Infiniband, but between the MICs themselves is 10GE.

Important Note: These machines are experimental, to test and port applications to this new platform. Success is not guaranteed for production jobs and many applications have not been ported yet to this architecture. The Support Team will do their best to solve any issues that may appear but it's not guaranteed.

Running modes

The Intel Xeon Phi accelerators provide different operation modes for accelerating computation:

- Native: the MIC is used like a node with more processors than usual. It requires specific compilation flags for the applications that will be run in them.
- Offload: the host binary executes some calculations on the mic. This is the mode Intel Math Kernel Libraries (MKL) use to speedup calculations of programs run where MICs are available.
- Mixed: Some processes run natively on the MIC and communicate with processes running natively on the host or hosts (MPI).

Connecting to the MIC

To access the MIC you first need to request one of the host nodes. You may request interactive access through queue 'mic-interactive' as follows:

```
bsub -Is -q mic-interactive -W 01:00 /bin/bash
```

Once connected to the host, you may ssh to either one of the mics:

```
ssh $HOSTNAME-mic0
ssh $HOSTNAME-mic1
```

You may compile on either the host, the logins or the MICs themselves.

Compiling for the MIC (Native and Mixed modes)

Compilation for MICs is only available with Intel Compilers suite. You must use the same compiler suite version with both the host program and the MIC program. The steps are the following:

- Load the environment for compilation

```
module purge
source /opt/intel/impi/4.1.1.036/bin64/mpivars.sh
source /opt/intel/2013.5.192/composer_xe_2013.5.192/bin/compilervars.sh intel64
export I_MPI_MIC=enable
```

- Compile using mpiicc or mpiifort for the host and the mic

```
mpiicc prog.c -o prog.host
mpiicc -mmic prog.c -o prog.mic
```

Executing interactively with the MIC

To manually send executions to the MIC you need to specify the bootstrap method for the mpirun connection:

```
export I_MPI_HYDRA_BOOTSTRAP=ssh
```

The full setup from a fresh connections is:

```
source /opt/intel/impi/4.1.1.036/bin64/mpivars.sh
export I_MPI_MIC=enable
export I_MPI_HYDRA_BOOTSTRAP=ssh

mpirun -genv I_MPI_DEVICE ssm ...
```

Examples

MPI between host and MIC (Mixed mode)

This example uses both the host and the MICs in native form but it's limited to just one node.

```
#!/bin/bash
#BSUB -n 2
#BSUB -x
#BSUB -oo %J.out
#BSUB -eo %J.err
#BSUB -q mic
#BSUB -W 00:10

source /opt/intel/impi/4.1.1.036/bin64/mpivars.sh
source /opt/intel/2013.5.192/composer_xe_2013.5.192/bin/compilervars.sh intel64
export I_MPI_MIC=enable

mpirun -genv I_MPI_DEVICE ssm -n 2 -host $HOSTNAME prog.host \
   : -n 244 -host $HOSTNAME-mic0 prog.mic \
   : -n 244 -host $HOSTNAME-mic1 prog.mic
```

By tweaking the -n value you control the number of processes per MIC.

MPI between MICs (Mixed mode)

The wrapper script bsc_micwrapper has been developed to ease the utilization of MPI across different hosts and their MICs. Check BSC Commands (section 6.4) for more information. Usage:

```
Params
 $1   : Number of MPI processes to execute in each host
 $2   : Number of MPI processes to execute in each mic of each host
 $3   : binary file *
```

*since host and mic binary are different, you must have <binary> and <binary>.mic in the same folder somewhere under /gpfs/scratch filesystem

Example:

```
#!/bin/bash
#BSUB -n 64
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -J Test
#BSUB -W 02:00
#BSUB -q mic

# The job will be allocated in 4 Xeon Phi nodes
# this will submit 16 MPI tasks in each node (host) and
# 240 MPI tasks in each mic (mic0 and mic1) of the 4 nodes allocated
# note that both
#    /gpfs/scratch/xxx/$USER/MIC/test_MPI_impi
#    /gpfs/scratch/xxx/$USER/MIC/test_MPI_impi.mic
# must exist

bsc_micwrapper 16 240 /gpfs/scratch/xxx/$USER/MIC/test_MPI_impi
```

5.7 MareNostrumSMP

Apart from the normal computation nodes, MareNostrum has available a special shared memory node. To check if you have been granted access to this special node check the output of bsc_queues. Check BSC Commands (section 6.4) for more information.

This node has the following specifications:

- 8 processors E7–8850 (10 Westmere-EX cores each) 2GHz 24MB L3
- 2 TB main memory
- 2 SSD local space (800 GB)
- 1 Inifiniband FDR10 card
- 2 10GE ports for GPFS access

To use it you just have to specify the queue as smp. For example:

```
#BSUB -q smp
```

The processor architecture of this node is different from normal MareNostrum nodes and does not support AVX instruction set. Therefore you may need to recompile your applications before using them on this node. Contact us (chapter 7) if you require assistance to port, compile or use your applications on this special node.

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 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
```

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 "-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
```

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/13.0.1
```

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 BSC Commands

The Support team at BSC has provided some commands useful for user's awareness and ease of use in our HPC machines. These commands are available through a special module (*bsc/current*) loaded at the beginning of any session. A short summary of these commands follows:

- bsc_acct: Displays accounting information on the project's allocation usage.
- bsc_jobcheck: Returns a comprehensive description of the resources requested by a jobscript.
- bsc_jobs: Show a list of your submitted jobs and those from other members of your group.
- bsc_load: Show the performance and resources usage of all the nodes of a specific running job.
- 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.
- bsc_micwrapper: Automatized script to execute MPI jobs using the Intel Xeon Phi Accelerators.

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

```
% man bsc_commands
% man bsc_jobs
% man bsc
```

6.5 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[5] available for new users. Further documentation and tutorials can be found on their website[6] or in the cluster at:

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

[5] https://www.bsc.es/support/TotalView-QuickView.pdf
[6] http://www.roguewave.com/products/totalview.aspx

6.6 Tracing jobs with BSC Tools

In this section you will find an introductory guide to get execution traces in Marenostrum. 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[7]

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

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

This package is compatible with the default MPI runtime in Marenostrum (OpenMPI). Packages corresponding to older versions and enabling compatibility with other MPI runtimes (IntelMPI, 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
#BSUB -n 128
#BSUB -o output_%J.out
#BSUB -e output_%J.err
#BSUB -R "span[ptile=16]"
#BSUB -J  job_name
#BSUB -W 00:10

module load extrae

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

Example for threaded (OpenMP or pthreads) jobs:

```
#!/bin/bash
#BSUB -n 1
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J job_name
#BSUB -W 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:

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
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

* 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.

7 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
```

7.1 Frequently Asked Questions (FAQ)

You can check the answers to most common questions at BSC's Support Knowledge Center[8]. 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][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][http://www.putty.nl]. Otherwise, any client compatible with SSH version 2 can be used.

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

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

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:

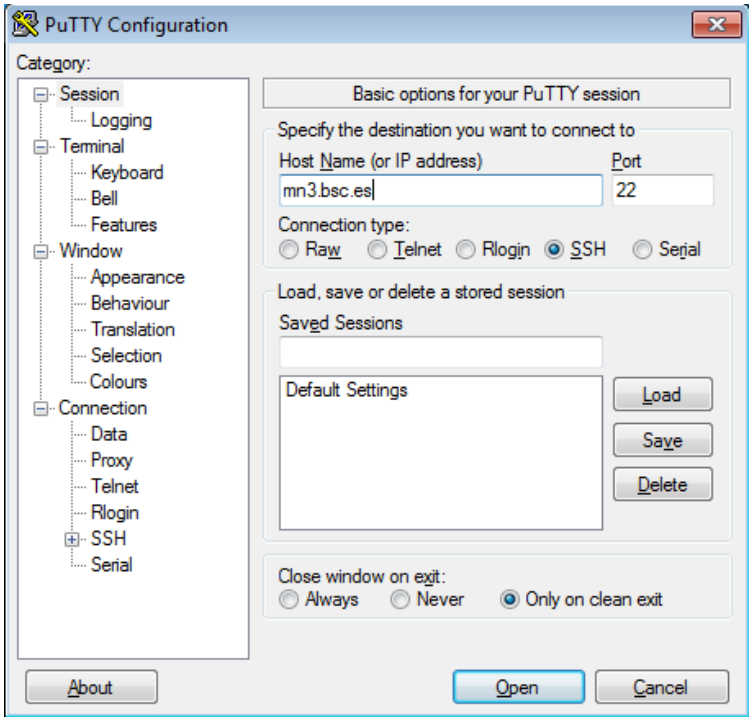

Figure 1: Putty client

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.

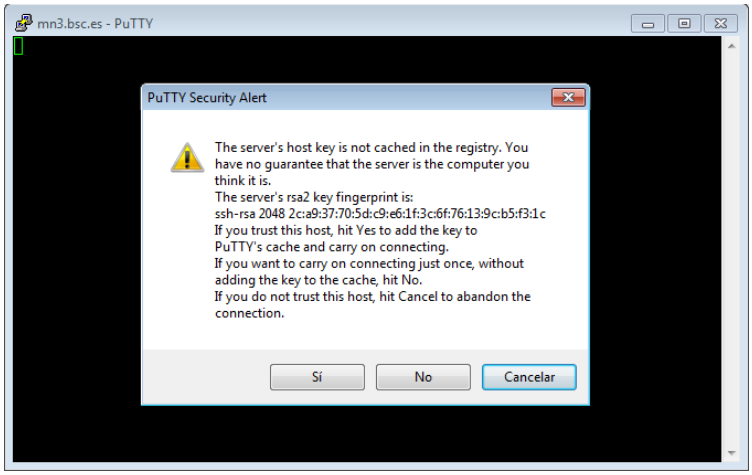

Figure 2: Putty certificate security alert

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:

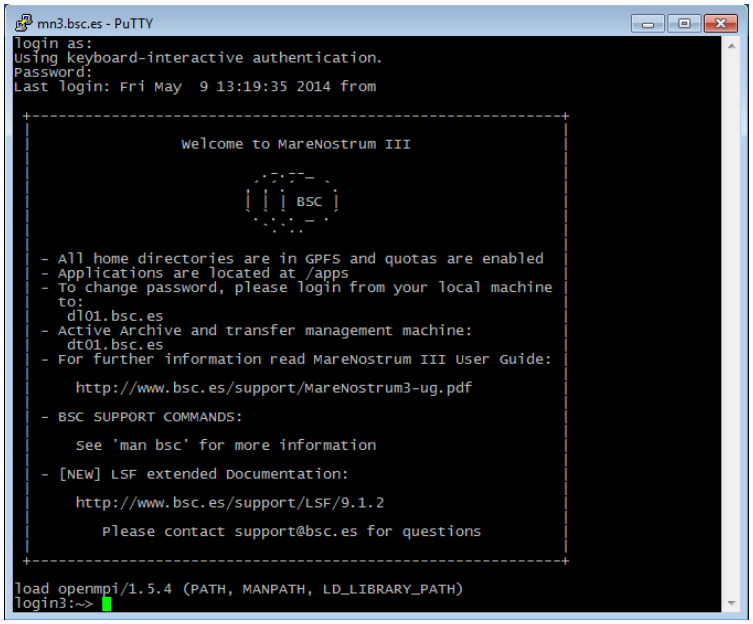

Figure 3: Cluster login

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[9]).

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 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 (mt1.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@mt1.bsc.es:remote_file local_file
Copy a file to the cluster:
> pscp.exe local_file username@mt1.bsc.es:remote_file
```

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

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:

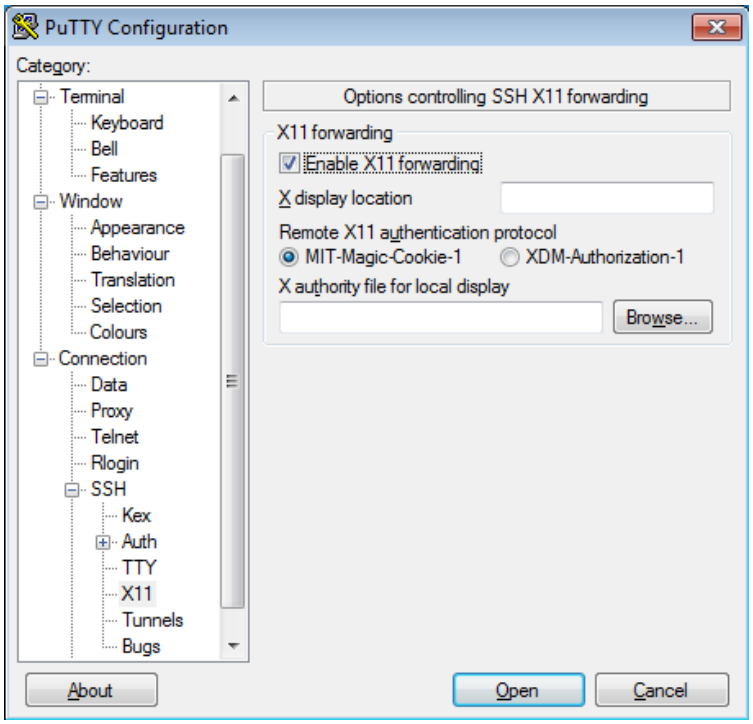

Figure 4: Putty X11 configuration