MareNostrum4 User's Guide

Barcelona Supercomputing Center

April 3, 2019

Contents

1 Introduction

This user's guide for the MareNostrum4 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

MareNostrum4 is a supercomputer based on Intel Xeon Platinum processors from the Skylake generation. It is a Lenovo system composed of SD530 Compute Racks, an Intel Omni-Path high performance network interconnect and running SuSE Linux Enterprise Server as operating system. Its current Linpack Rmax Performance is 6.2272 Petaflops.

This general-purpose block consists of 48 racks housing 3456 nodes with a grand total of 165,888 processor cores and 390 Terabytes of main memory. Compute nodes are equipped with:

- 2 sockets Intel Xeon Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of **48 cores per node**
- L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K
- 96 GB of main memory **1.880 GB/core**, 12x 8GB 2667Mhz DIMM (216 nodes high memory, 10368 cores with 7.928 GB/core)
- 100 Gbit/s Intel Omni-Path HFI Silicon 100 Series PCI-E adapter
- 10 Gbit Ethernet
- 200 GB local SSD available as temporary storage during jobs ($TMPDIR=/scratch/tmp/[jobid])

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

Remember that the BIOS and kernel reserves memory, so the actual total usable RAM that commands like "free" or "lstopo" report will be slighlty lower than 96GB (∼94GB).

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/2017.4*

Intel Compilers

The latest Intel compilers provide the best possible optimizations for the Xeon Platinum 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/2017.4), the Intel MPI software stack (impi/2017.4) and the math kernel libraries MKL (mkl/2017.4) 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/2017.4
```

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.

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

2.2 Intel Compiler Licences

For reasons of licensing we recommend compiling using either login1 or the partition *interactive*. We currently have node locked licences installed that allow unlimited compilations in the machines login1 and the ones available via the queue interactive (logins 4 and 5). To compile in the rest of the compute and login nodes we have a limited amount of floating licences available. Should all of them be in use when trying to compile, you will experience a delay when the compiler starts and tries to checkout a licence.

In this case an error message like the one below will appear. In this case please switch to Login 1, Login 4 or Login 5 to compile without limitations and licencing issues.

```
ifort: error #10052: could not checkout FLEXlm license

Error: A license for Comp-CL is not available (-9,57).

License file(s) used were (in this order):
   1.  Trusted Storage
**  2.
  /gpfs/apps/MN4/INTEL/2017.4/compilers_and_libraries_2017.4.196/linux/bin/intel64/../../
   Licenses
**  3.  /home/bsc18/bsc1888//Licenses
**  4.  /opt/intel/licenses
**  5.  /Users/Shared/Library/Application Support/Intel/Licenses
**  6.
  /gpfs/apps/MN4/INTEL/2017.4/compilers_and_libraries_2017.4.196/linux/bin/intel64/license.lic
**  7.
  /gpfs/apps/MN4/INTEL/2017.4/compilers_and_libraries_2017.4.196/linux/bin/intel64/
   login1_COM_L___1.lic
**  8.
  /gpfs/apps/MN4/INTEL/2017.4/compilers_and_libraries_2017.4.196/linux/bin/intel64/
   login4_COM_L___1.lic
**  9.
  /gpfs/apps/MN4/INTEL/2017.4/compilers_and_libraries_2017.4.196/linux/bin/intel64/
   login5_COM_L___1.lic

Please refer http://software.intel.com/sites/support/ for more information..
```

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 MareNostrum is GCC 7.1.0

```
module load gcc/7.1.0
```

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

2.3 Login Nodes

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

```
mn1.bsc.es
mn2.bsc.es
mn3.bsc.es
```

Note: Due to the upgrade of MareNostrum the actual login nodes and their configuration changed. If you had access to the system before and the SSH identification of the logins stored in your known_hosts file you might receive the following warning:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    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.
```

```
The fingerprint for the RSA key sent by the remote host is
SHA256:sLS2WHI4/tz/dlKt+9Pu9vUcD4HFl/Gg/fGRjNrEkjc.
Please contact your system administrator.
Add correct host key in /home/mlopez12/.ssh/known_hosts to get rid of this message.
Offending RSA key in /home/mlopez12/.ssh/known_hosts:73
  remove with:
  ssh-keygen -f "/home/mlopez12/.ssh/known_hosts" -R mn1.bsc.es
RSA host key for mn1.bsc.es has changed and you have requested strict checking.
Host key verification failed.
```

This might happen the first time you connect. To solve this issue follow the hint given to remove the offending key by executing the following command

```
ssh-keygen -f "/home/mlopez12/.ssh/known_hosts" -R mn1.bsc.es
```

Afterwards you will have the new key stored and the warning should not appear again.

2.4 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.

2.5 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 8), 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. To get access to this service it's required to contact "support@bsc.es" requesting its use, providing the local IP of the machine from where it will be used.

The selected data transfer tool is Globus/GridFTP[2] 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:

[2] 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[3]

2.6 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 2.5) (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/
```

[3] 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 2.5). Therefore if you wish to navigate through AA directory tree you have to login into dt01.bsc.es

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 and Data Transfer Machine (section 2.5)
- *Local hard drive*: Every node has an internal hard drive

3.1 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 3.3).

3.2 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 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.
- */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 3.4) 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 3.4) 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 3.4) per group will be enforced depending on the space assigned.

3.3 Local Hard Drive

Every node has a local solid state (SSD) hard drive (HDD) 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 200 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.**

3.4 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 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 7).

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.

Important notice: All jobs requesting 48 or more cores will automatically use all requested nodes in exclusive mode. For example if you request 49 cores you will receive two complete nodes (48 cores * 2 = 96 cores) and the consumed runtime of these 96 cores will be reflected in your budget.

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
```

The standard configuration and limits of the queues are the following

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 7).

Queue	Maximum numer of nodes (cores)	Maximum wallclock
Debug	16 (768)	2 h
Interactive	(max 4 cores)	2 h
BSC	50 (2400)	48 h
RES Class A	200 (9600)	72 h
RES Class B	200 (9600)	48 h
RES Class C	32 (1536)	24 h
PRACE	400 (19200)	72 h

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 in the interactive partition has to be done through SLURM:

```
salloc --partition=interactive
or
salloc -p interactive
```

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=debug
```

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 --constraint=highmem
```

Select which configuration to run your job on, for example "highmem" to run the job on a HighMem node with 7928 MB per core. Without this directive the jobs will be sent to standard nodes that have 1880 MB of RAM per core. There are only a limited number of high memory nodes available, 216 nodes (10368 cores) out of 3456 nodes (165888 cores) in total. Therefore when requesting these nodes you can expect **significantly longer queueing times** to fulfil the resource request before your job can start.

The accounting for one core hour in standard and highmem nodes is the same, e.g. 1 core hour per core per hour will be budgeted. For faster turnaround times in the queues you can also use standard nodes and run less processes per node. For this you will need to use more cores per task, as every cores requested comes with its 2 GB RAM. You can do this by specifying the flag *#SBATCH –cpus-per-task=number* and your budget will **get charged for all cores requested**.

```
#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 --cpu-freq=<number>
```

Request that job steps initiated by srun commands inside this sbatch script be run at some requested frequency if possible, on the cores selected for the step on the compute node(s). Available frequency steps: 2.10 GHz, 2.00 GHz, 1.90 GHz, 1.80 GHz, 1.70 GHz, 1.60 GHz, 1.50 GHz, 1.40 GHz, 1.30 GHz, 1.20 GHz, 1.10 GHz, 1 GHz

The value is being given in KHz and therefore you will need to specify the values from 2100000 to 1000000.

By default both the turbo boost an speed step technologies are activated in MareNostrum4.

```
#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.

For more information:

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

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

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 MN4 node using 48 cores on the debug queue:

```
#!/bin/bash
#SBATCH --job-name=omp
#SBATCH --workdir=.
#SBATCH --output=omp_%j.out
#SBATCH --error=omp_%j.err
#SBATCH --cpus-per-task=48
#SBATCH --ntasks=1
#SBATCH --time=00:10:00
#SBATCH --qos=debug
./openmp_binary
```

- Running on two MN4 nodes using a pure MPI job

```
#!/bin/bash
#SBATCH --job-name=mpi
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=96
srun ./mpi_binary
```

- Running a hybrid MPI+OpenMP job on two MN4 nodes with 24 MPI tasks (12 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=24
#SBATCH --cpus-per-task=4
#SBATCH --tasks-per-node=12
#SBATCH --time=00:02:00
srun ./parallel_binary> parallel.output
```

- Running on four high memory MN4 nodes with 1 task per node, each using 48 cores:

```
#!/bin/bash
#SBATCH --job-name=test_parallel
#SBATCH --workdir=.
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=48
#SBATCH --tasks-per-node=1
#SBATCH --time=00:02:00
#SBATCH --constraint=highmem
srun ./parallel_binary> parallel.output
```

4.6 Resource usage and job priorities

Projects will have assinged a certain amount of **compute hours or core hours** that are available to use. One core hour is the computing time of one core during the time of one hour. That is a full node with 48 cores running a job for one hour will use up 48 core hours from the assigned budget. The accounting is solely based in the amount of compute hours used.

The **priority of a job** and therefore its scheduling in the queues is being determined by a multitude of factors. The most important and influential ones are the fairshare in between groups, waiting time in queues and job size. MareNostrum is a system meant for and favouring large executions so that jobs using more cores have a higher priority. The time while waiting in queues for execution is being taken into account as well and jobs gain more and more priority the longer they are waiting. Finally our queue system implements a fairshare policy between groups. Users who did not run many jobs and consumed compute hours will get a higher priority for their jobs than groups that have a high usage. This is to allow everyone their fair share of compute time and the option to run jobs without one group or another being favoured. You can review your current fair share score using the command

```
sshare -la
```

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 7)).

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/2017.4
```

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.

Module custom stack size

The stack size is 2GB by default, but there are modules that can change that value when they are loaded. That's to prevent errors that would happen otherwise (for example, while using python threads). Here's a list of the affected modules:

- python/3-intel–2018.2 (64 MB)
- python/2-intel–2018.2 (64 MB)
- python/2.7.13 (10 MB)
- python/2.7.13_ML (64 MB)
- python/2.7.14 (10 MB)
- python/3.6.1 (10 MB)
- python/3.6.4_ML (64 MB)
- paraview/5.4.0 (64 MB)
- paraview/5.5.2 (64 MB)
- vmd/1.9.3 (64 MB)
- vmd/1.9.3-python (64 MB)
- igv/2.3.94 (64 MB)

You can check your stack size at any time using the command:

```
% ulimit -s
```

If by any chance this stack size doesn't fit your needs, you can add to your job script a command to have a custom set value after the module load commands. This way, you can have an appropiate stack size while using the compute nodes:

```
% ulimit -Ss <new_size_in_KB>
```

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.
- bsc_acct: Displays accounting information on the project's allocation usage.
- bsc_load: Displays job load information across all related computing nodes.

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

```
% man bsc_commands
```

6 PRACE

A large part of MareNostrums capacities is reserved for users of the PRACE Research Infrastructure[4]. There are regular calls ongoing to gain access to the system via this way and more information can be found here[5].

6.1 Compute hour allocation

PRACE projects that have been awarded compute hours are expected to **start their work and the consumption of hours immediately when the project begins** The allocated hours are meant to be **consumed consistently and proportionally** during the projects lifetime. To aid users of the system we will regularly inform those groups by mail that are exhibiting a usage below the proportional average for their project.

To avoid the accumulation of too many hours towards the end of the project when it might be complicated to use them up fully we implement a **sliding usage window of 3 months**. This means that after 3 months the hours of the first month that have not been consumed will count as used up and deducted from the total of available compute hours. The sliding-window is moved every 3 months, so we will deduct hours on the month 3rd, 6th and 9th of the project.

> For example a 12 months project has been awarded with 12 Mio core hours and starts in January. The monthly proportional consumption is therefore 1 Mio core hours. Due to a sabbatical of the PIs group in Menorca the actual usage of the project only starts in *April*. Therefore *1 Mio core hours* from the month of *January* will be deducted from the total assignation. If the group continues to consume normally the remaining 11 Mio core hours can be used up entirely.

This regulation does not affect overconsumption of the awarded compute hours. Therefore you could also use up all of your hours within the first month of the project.

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
```

[4] http://www.prace-ri.eu/

[5] http://www.prace-ri.eu/call-announcements/

7.1 Frequently Asked Questions (FAQ)

You can check the answers to most common questions at BSC's Support Knowledge Center[6]. There you will find online and updated versions of our documentation, including this guide, and a listing with deeper answers to the most common questions we receive as well as advanced specific questions unfit for a general-purpose user guide.

8 Appendices

8.1 SSH

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

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

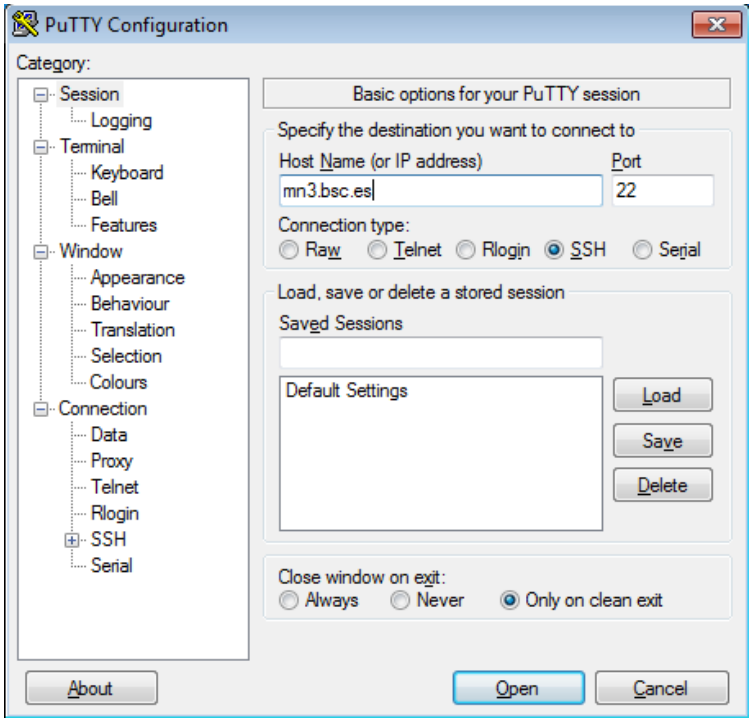

Figure 1: Putty client

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.

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

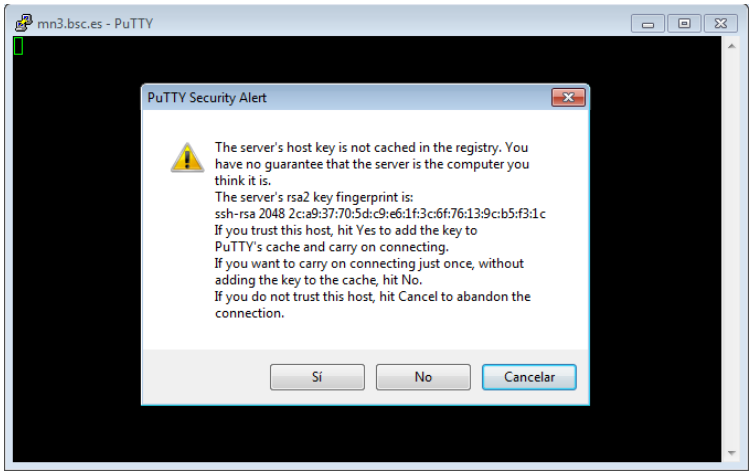

Figure 2: Putty certificate security alert

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

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

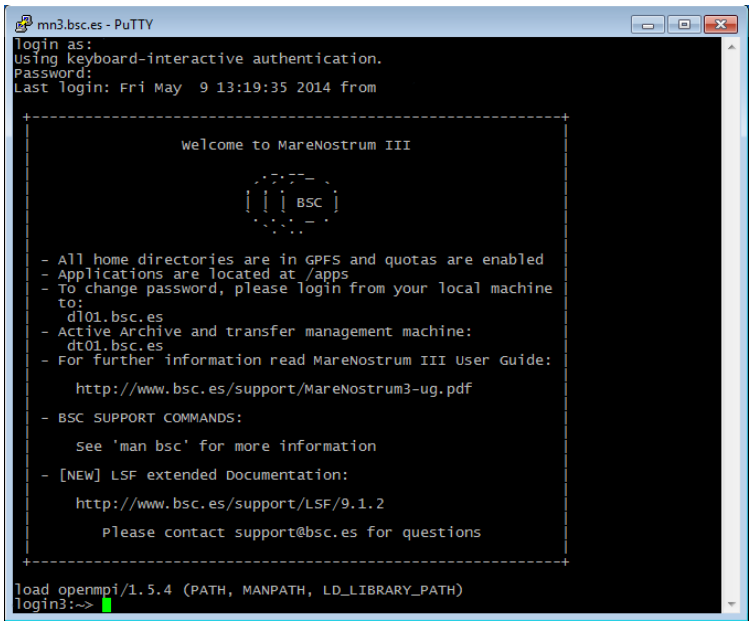

Figure 3: Cluster login

8.2 Transferring files

To transfer files to or from the cluster you need a secure ftp (sftp) o secure copy (scp) client. There are several different clients, but as previously mentioned, we recommend using of Putty clients for transferring files: **psftp** and **pscp**. You can find it at the same web page as Putty (http://www.putty.org[7]).

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)

[7] 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
```

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:

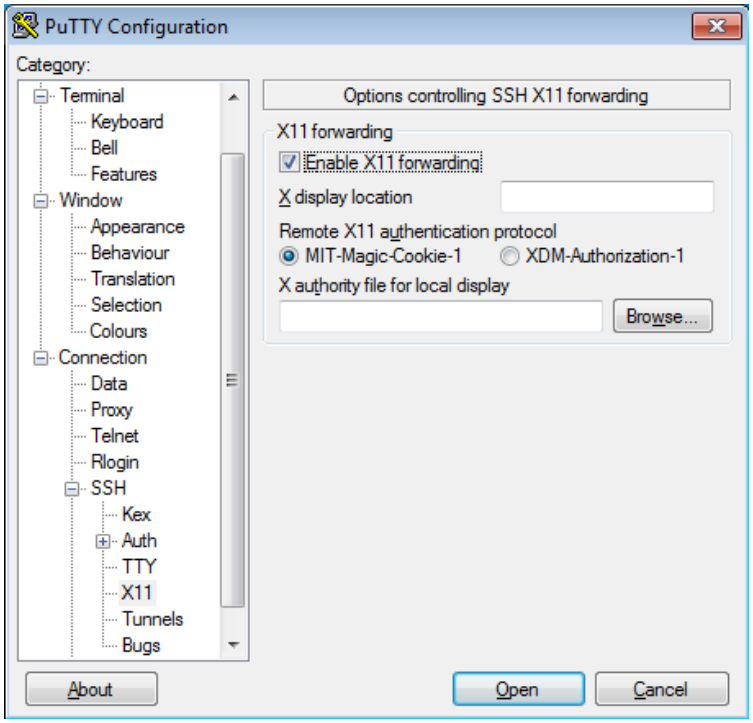

Figure 4: Putty X11 configuration