
BSC Altix 2 UV100 User's Guide



Barcelona Supercomputing Center

Copyright © 2011 BSC-CNS

Table of Contents

1. Introduction	1
2. System Overview	1
3. Connecting to Altix2	2
3.1. Password Management	2
3.2. Transferring files	3
3.3. HSM Management	4
3.4. Graphical applications	4
4. Running Jobs	4
4.1. Submitting jobs	4
5. Software Environment	7
5.1. C Compilers	7
5.2. FORTRAN Compilers	8
5.3. Software in Altix 2	9
5.4. Modules Environment	9
6. Getting help	11
A. SSH	11

1. Introduction

This user's guide for the Altix UV100 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.

We hope you can find most of the information you need to use our computing resources and technical documentation about the machine. Please read carefully this document and if any doubt arises don't hesitate to contact our support group at <support@bsc.es>

2. System Overview

Altix UV100 is a shared memory machine, with a cc-NUMA architecture (Cache Coherent Non-Uniform Memory Access). The total amount is 96 CPUs with 1.5TB of RAM memory composed by:

- 3 IRUs, which are the basic SGI unit interconnected with Numalink 5 at 15GB/s
- Each IRU has 2 blades IP93
- Each IP93 blade has:
 - 2 Nehalem-EX Intel(R) Xeon(R) CPU E7-8837 8-core @ 2.67GHz
 - 16 DIMM of 16GB RAM DDR3 @ 1066MHz, with 256GB of total memory per blade

3. Connecting to Altix2

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:

```
bscsm02.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 Appendix A.

Here you have an example of logging into the cluster from a UNIX environment:

```
> ssh -l username bscsm02.bsc.es
username's password:

Last login: Mon Jan  9 08:17:37 2012 from XXXX

username@bscsm02 ~]$_
```

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

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

The first time that you connect to the cluster, secure shell needs to interchange some initial information to establish the communication. This information consists of the acceptance of the RSA key of the remote host, you must answer 'yes' or 'no' to confirm the acceptance of this key.

If you cannot get access to the system after following this procedure, first consult Appendix A for extended information about Secure Shell, or you can contact us, (see section 9 to know how to contact with us).

Once inside 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 have to learn the basics before you could do 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.
```

Take into account that the password change is not instantaneous, it takes about 10 minutes to be effective. This password is shared between all the GPFS cluster machines.

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

3.2.1. 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@bscsmp02.bsc.es:
username's password:
```

```
localsystem$ sftp username@bscsmp02.bsc.es
username's password:
sftp> put localfile
```

These are the ways to retrieve files from Altix 2 to your local machine:

```
localsystem$ scp username@bscsmp02.bsc.es:remotefile localdir
username's password:
```

```
localsystem$ sftp username@bscsmp02.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.

3.2.2. Data Transfer Machine

There is a new machine dedicated to Data Transfers. This machine is accessible through ssh with the same username and password than Altix 2.

```
> ssh -l username dt01.bsc.es
username's password:

Last login: Fri Sep 2 15:07:04 2011 from XXXX

username@dtransfer1:~>
```

As it was said before, this machine shares the entire GPFS filesystem with Altix 2 and other machines. Besides scp and sftp, this machine allows some other useful transfer protocols:

- BBCP
- GRIDFTP
- FTPS

3.3. HSM Management

To move or copy from/to HSM 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. (mnq)

`dtcalcel` works like `mncancel` (see below) for transfer jobs.

It is important to note that these kind of jobs can be submitted from both 'Altix UV100' (automatic file management within a production job) and 'dt01.bsc.es' machine. HSM is not mounted in Altix UV100 machine. Therefore if you wish to navigate through HSM directory tree you have to login dt01.bsc.es

3.4. Graphical applications

You can execute graphical applications. To do that the only way is tunneling all the graphical traffic through the established Secure shell connection.

You will need a Xserver running on your local machine to be able to show the graphical information. Most of the UNIX flavors have an X server installed by default. In a Windows environment, you will probably need to download and install some type of X server emulator. (see appendix A)

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

Here you have an example:

```
localsystem$ ssh -X -l username bscsmp02.bsc.es
username's password:

Last login: Mon Jan  9 08:17:37 2012 from XXXX

username@bscsmp02 ~]$
```

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

4. Running Jobs

Slurm is the utility used at Altix 2 UV100 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.

4.1. Submitting jobs

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

4.1.1. SLURM wrapper commands

These are the basic directives to submit jobs:

- `mnsuubmit <job_script>`

submits a “job script” to the queue system (see below for job script directives).

- `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.

4.1.2. 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, with the following syntax:

```
# @ directive = value
```

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

```
# @ class = class_name
```

The partition where the job is to be submitted. Let this field empty unless you need to use "interactive" or "debug" partitions.

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

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

```
# @ initialdir = pathname
```

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

```
# @ error = file
```

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

```
# @ output = file
```

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

```
# @ total_tasks = number
```

The number of processes to start.

Optionally, you can specify how many threads each process would open with the directive:

```
# @ cpus_per_task = number
```

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

```
# @ tasks_per_node = number
```

The number of tasks assigned to a node.

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

Table 1. SLURM environment variables

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

4.1.3. 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:

```
> mns submit 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
# @ cpus_per_task     = 1
# @ wall_clock_limit = 00:02:00

mnr run ./parallel_binary > parallel.output
```

```
#!/bin/bash
# @ job_name          = test_parallel
# @ initialdir        = .
# @ output            = mpi_%j.out
```

```
# @ error          = mpi_%j.err
# @ total_tasks    = 1
# @ cpus_per_task  = 12
# @ wall_clock_limit = 00:02:00

export OMP_NUM_THREADS = 12
./parallel_binary > parallel.output
```

5. Software Environment

5.1. C Compilers

In the cluster you can find these C/C++ compilers :

icc / icpc -> Intel C/C++ Compilers version 12.0

```
man icc
man icpc
```

gcc /g++ -> GNU Compilers for C/C++, Version 4.3.4

```
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 Altix 2.

Table 2. Data types

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

5.1.1. Distributed Memory Parallelism

To compile MPI programs it is recommended to use the following handy wrappers: `mpicc`, `mpicxx` for C and C++ source code. 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
```

5.1.2. 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.

```
usertest@bscsmp02:~> icc -openmp -o exename filename.c
usertest@bscsmp02:~> icpc -openmp -o exename filename.C
```

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

5.1.3. 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 Altix 2 UV100 you can find these compilers :

`ifort` -> Intel Fortran Compilers 12.0

```
man ifort
```

`gfortran` -> GNU Compilers for FORTRAN, Version 4.3.4

```
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"`.

5.2.1. 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
```

5.2.2. Shared Memory Parallelism

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

```
% ifort -openmp
```

5.2.3. Automatic Parallelization

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

```
% ifort -parallel
```

5.3. Software in Altix 2

All software and numerical libraries available at the cluster can be found at /apps. If you need something that it is not there, you may contact support@bsc.es to get it installed.

5.4. Modules Environment

The Environment Modules package 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.

<http://modules.sourceforge.net/>

A coherent set of software packages is divided into five categories:

- Environment: modulefiles dedicated to prepare the environment, for example, get all necessary variables to use mpibull to compile and even run programs
- Tools: useful tools which can be used at any time (php, perl, ...)
- Applications: High Performance Computers programs (GROMACS, ...)
- Libraries: Those are typically loaded at a compilation time, they load into the environment the correct compiler and linker flags (FFTW, LAPACK, ...)
- Compilers: You can use different compilers and versions in this package (intel, gcc, ...)

5.4.1. Modules tool usage

This section will explain a Quick Guide for the Modules environment usage at Altix 2 UV100.

If you run "module help", you will be able to see all module commands. More information in module(1) manpage.

```
Modules Release 3.1.6 (Copyright GNU GPL v2 1991):
Available Commands and Usage:
+ add|loadmodulefile [modulefile ...]
+ rm|unloadmodulefile [modulefile ...]
+ switch|swapmodulefile1 modulefile2
+ display|showmodulefile [modulefile ...]
+ avail[modulefile [modulefile ...]]
+ use [-a|--append]dir [dir ...]
+ unusedir [dir ...]
+ update
+ purge
+ list
+ clear
+ help[modulefile [modulefile ...]]
+ whatis[modulefile [modulefile ...]]
+ apropos|keywordstring
+ initaddmodulefile [modulefile ...]
```

```

+ initprependmodulefile [modulefile ...]
+ initermmodulefile [modulefile ...]
+ initswitchmodulefile1 modulefile2
+ initlist
+ initclear

```

The most important commands are: list, avail, load, unload, switch and purge:

- **module list** shows all the modules you have loaded:

```

% module list

Currently Loaded Modulefiles:
1) intel/12.0.0          2) mpt/2.04
3) compilewrappers/yes

```

- **module avail** shows all the modules that user is able to load:

```

% module avail

-----
/usr/share/modules/modulefiles
-----
MPInside/3.1          module-cvs          mpiplace/1.01
perfboost            sgi-upc/1.04
chkfeature           module-info         mpt/2.04
perfcatcher          sgi-upc-devel/1.04
dot                  modules              null
scotch/5.1.11        use.own

-----
/apps/modules/modulefiles/applications
-----
AMBER/11(default)  NAMD/2.8(default)

-----
/apps/modules/modulefiles/compilers
-----
gcc/4.6.1(default)  intel/12.0.0(default)

-----
/apps/modules/modulefiles/environment
-----
compilewrappers/yes(default)  openmpi/1.4.4
openmpi/1.5.4(default)

-----
/apps/modules/modulefiles/libraries
-----
blas/1.0(default)      fftw/3.3(default)
mkl/10.3.0(default)    fftw/3.2.2
lapack/3.4.0(default)  scalapack/1.8.0(default)

-----
/apps/modules/modulefiles/tools
-----

```

- **module load** let user load the necessary environment variables for the selected modulefile (PATH, MANPATH, LD_LIBRARY_PATH...)

```
% module load fftw
load fftw/3.3 (CFLAGS,FFLAGS,LDFLAGS)
```

- **module unload** removes all environment changes made by module load command:

```
% module unload fftw
remove fftw/3.3 (CFLAGS,FFLAGS,LDFLAGS)
```

- **module switch** acts as module unload and module load command at the same time:

```
% module load fftw
load fftw/3.3 (CFLAGS,FFLAGS,LDFLAGS)
% module switch fftw fftw/3.2.2
switch1 fftw/3.3 (CFLAGS,FFLAGS,LDFLAGS)
switch2 fftw/3.2.2 (CFLAGS,FFLAGS,LDFLAGS)
switch3 fftw/3.2.2 (CFLAGS,FFLAGS,LDFLAGS)
ModuleCmd_Switch.c(243):VERB:4: done
% module list
Currently Loaded Modulefiles:
  1) /fftw/3.2.2
```

- **module purge** removes all the changes made by previous commands:

```
% module list
Currently Loaded Modulefiles:
  1) intel/12.0.0          2) mpt/2.04
  3) compilewrappers/yes
% module purge
remove intel/12.0.0 (PATH, MANPATH, LD_LIBRARY_PATH)
remove mpt/2.04 (CFLAGS,FFLAGS,LDFLAGS)
```

6. Getting help

In addition to our online information tools described in section 3, 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

A. 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. BSC

recommend the use of OpenSSH client that can be download from <http://www.openssh.org>, but any client compatible with SSH version 2 can be used.

In windows systems BSC recommends the use of putty. It is a free SSH client that you can download from <http://www.putty.nl/>. But you can also, any client compatible with SSH version 2 can be used.

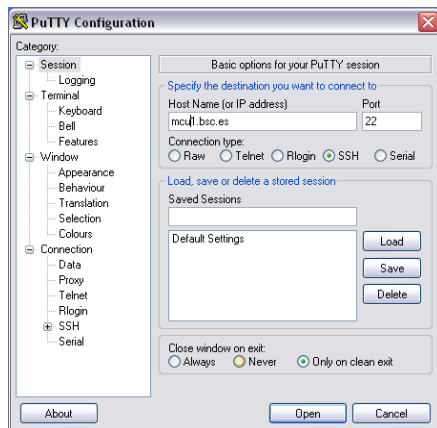
In the next lines we will describe how to install, configure and use a ssh client under Windows systems.

Once the client has been installed, it is needed to fix some settings. At least it is necessary to

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

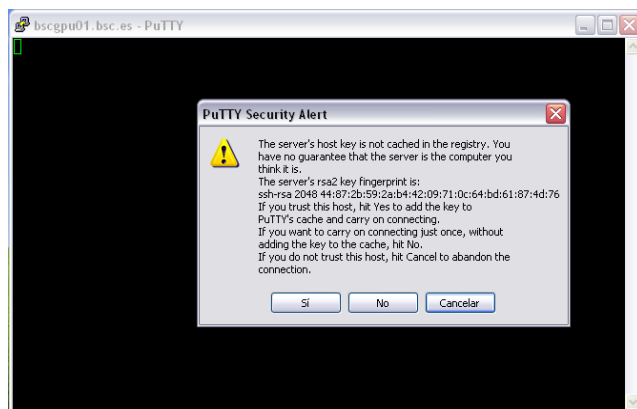
For example with putty client:

Figure A.1. Putty client



This is the first windows 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 A.2. Putty certificate security alert



IMPORTANT: If you see this warning another time and you haven't modified or reinstalled the ssh client , please, don't log in , and contact support@bsc.es.

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

Figure A.3. Cluster login

```

bsc99186@nrvb127:~
login as: bsc99186
Access denied
bsc99186@nrvb127:~$ ssh bsc99186@nrvb127
Last login: Fri Sep  9 09:53:03 2011 from 64.86.53.46

-----
                Welcome to BSC GPU Cluster
-----
- All home directories are in GPFS and quotas are enabled
- Applications are located at /gpfs/apps/NVIDIA

Please contact support@bsc.es for questions
-----

[bsc99186@nrvb127 ~]$

```

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

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>)

Example A.1. 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 (bscsm01.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 Altix to your local machine.
- *put file_name* : To transfer a file from your local machine to Altix
- *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 Altix, and from Altix 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 Altix:

```
> pscp.exe username@10.3.231.21:remote_file local_file
```

Copy a file to Altix:

```
> pscp.exe local_file username@10.3.231.21:remote_file
```

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 A.4. Putty X11 configuration

