Altamira Users Guide

Introduction

This user's guide for the Altamira supercomputer 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 basic notions on scientific computing, in particular the basic commands of the Unix operating system, and also with basic techniques for the execution of applications in a supercomputer, like MPI or OpenMP.

The information in this guide includes basic technical documentation about the Altamira supercomputer, the software environment, and also on available applications.

Please read it carefully and if any doubt arises don't hesitate to contact our support mail (see below in Getting help).

Requesting an account

Altamira users include researchers at the University of Cantabria, researchers who get execution time at the Spanish Supercomputing Network (RES), and other researchers. The assignment of an account and execution time requires a request form, contact us **in case of doubt or for urgent requests.**

System Overview

Altamira includes **158 main compute nodes** and 1 **login** server.

Main compute nodes have two Intel Sandybridge E5-2670 processors, each one with 8 cores operating at 2.6 GHz and a cache of 20MB, 64 GB of RAM memory (i.e. 4 GB/core) and 500 GB local disk.

Operating System (OS)

Main compute and login nodes run **Centos 7.9**

Network

The internal network in Altamira includes:

- **Infiniband Network (FDR)**: High bandwidth network used by parallel applications communications and data transfer.
- **Gigabit Network**: Ethernet network used by the management services.

Storage

All the nodes are connected to a global storage system based on **GPFS** (Global Parallel File System) providing a total storage of **1.5 PB**.

File Systems

Each nodes has several areas of disk space for installing system or programs and 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 2 different types of storage available inside a node:

- **GPFS Filesystems**: GPFS is a distributed networked filesystem which can be accessed from all the nodes
- Local Hard Drive: Every compute node has an internal hard drive

GPFS filesystem

The IBM General Parallel File System (GPFS) is a high-performance shared-disk file system that can provide fast, reliable data access from all blades 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. These filesystems are the recommended to use with most jobs, because GPFS provides high-performance I/O by "striping" blocks of data from individual files across multiple disks on multiple storage devices and reading/writing these blocks in parallel. In addition, GPFS can read or write large blocks of data in a single I/O operation, thereby minimizing overhead.

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

- **/gpfs/users/res /home: users home:** This filesystem has the home directories of all the users, when you log into Altamira you start in your home directory by default. Every user will have their own home directory to store the executables, own developed sources and their personal data. Quotas are in effect that limit the amount of data that can be saved here, a default quota will be enforced to all users.
- **/gpfs/res_projects:** In addition to the home directory, there is a directory in /gpfs/res_projects for each group of users of Altamira. All the users of the same project will share their common /gpfs/res_projects space and it is responsibility of each project manager to determine and coordinate the better use of this space, and how it is distributed or shared between their users. If a project needs more disk space in this filesystem or in any other of the GPFS filesystems, the project manager has to make a request for this extra space needed, specifying the space needed and the reasons why it is needed (see Getting Help section to know how contact us).
- **/gpfs/res_scratch:** Each Altamira user will have a directory over /gpfs/res_scratch, you must use this space to store temporary files of your jobs during its execution. By default, files may reside for up to 7 days without modification in this filesystem, any older file might be removed. A quota per group will be enforced depending on the space assigned.
- **/gpfs/res_apps:** Over this filesystem will reside the applications and libraries that have already been installed on Altamira. Take a look at the directories or go to Software section to know the applications available for general use. To use an application, you must load the module as is detailed below in the Software section. Before installing any application that is needed by your project, first check if this application is already installed on the system. If some application that you need is not on the system, you will have to ask our user support team to install it. Check Getting Help Section how to contact with us. If it is a general application with no restrictions in his use, this will be installed over a public directory, that is over /gpfs/res_apps so all users on Altamira could make use of it. If the application needs some type of license and his use must be restricted, a private directory over /gpfs/res_apps will be created, so only the required users of Altamira could make use of this application. All applications on /gpfs/res_apps will be installed, controlled and supervised by the user support team. This doesn't mean that the users could not help in this task, both can work together to get the best result. The user support can provide his wide experience in compiling and optimizing applications in the Altamira cluster and the users can provide his knowledge of the application to be installed. All that general applications that have been modified in some way from its normal behavior by the project users' for their own study, and may not be suitable for general use, must be installed over /gpfs/res_projects or /gpfs/res_home depending on the usage scope of the application, but not over /gpfs/res_apps.

Batch System

The job scheduling system running at Altamira is **SLURM**. Current version of Slurm in Altamira is **20.11.7**.

Partition

Slurm has partitions to allocate jobs. In altamira, a group of nodes with same features are assigned to a partition. This following table resumes the main partitions and the principal properties of each of them. **By default, compute is configured if --partition option is not defined in the job. Set it to res if you are coming from a RES project** or use the testing partition for test jobs before submitting to the production partition (compute). Test jobs in the testing queue has a limit of 8 cpus and 3 hours of time as you can see below at qos.

Partition	Description
compute	Main partition in Altamira
res	Group of nodes for RES users
testing	This partition can be used to test new workflows and also to help new users to familiarise themselves with the SLURM batch system. Both serial and parallel code can be tested on the testing partition.

QoS

Specify the Quality of Service (Qos) for each job submitted to slurm. Jobs request a QOS using the "--qos=" option to the *sbatch*, *salloc*, and *srun* commands. By default,

- the **local** user is assigned to the **main** and **testing** queues.
- the users coming from a **RES** project belong to the **res_a** and **testing** queues.

Group	Queue	Description	Limit number of nodes	Max CPU cores per user	Max Run Time	Max jobs per user
Local	main	Main queue for local users	158	512	3 days	1000
	testing	This queue can be used to test new workflows and also to help new users to familiarise themselves with the SLURM batch system. Both serial and parallel code can be tested on the testing queue.	4	8	3 hours	2
Res	res_a	Main queue for RES users	128	1024 (64 nodes)	3 days	500
	res_c	Queue assigned for res users when they reach the period hours limits requested	4	64 (4 nodes)	1 day	10

Connecting to Altamira

Once you have a username and its associated password you can get into Altamira system, connecting the **only one login endpoint available at login1.ifca.es**. The password provided is temporal, you must change this initial password after connecting to IPA. Also use a strong password (do not use a word or phrase from a dictionary and do not use a word that can be obviously tied to your person).

You must use Secure Shell (ssh) tools to login into or transfer file into Altamira. We do not accept incoming connections from protocols as telnet, ftp, rlogin, rcp, or rsh commands. The connection to Altamira must be done in the following way:

$ ssh username@loginX.ifca.es (where X is 1 or 2)

Once you are logged into Altamira you cannot make outgoing connections for security reasons (contact us in an exceptional case).

If you cannot get access to the system after following this procedure you can contact us, (see Getting Help to know how to contact with us).

Login Node

Once inside the machine you will be presented with a UNIX shell prompt and you'll normally be in your home ($HOME) directory. If you are new to UNIX, you'll have to learn the basics before you could do anything useful.

The machine in which you will be logged in will be the login node of Altamira (login1). This machine acts as front end, and it is used typically for editing, compiling, preparation/submition of batch executions and as a gateway for copying data inside or outside Altamira.

It is not permitted the execution of cpu-bound programs on this node, if some compilation needs much more cputime than the permitted, this needs to be done through the batch queue system. It is not possible to connect directly to the compute nodes from the login node, all resource allocation is done by the batch queue system.

Compute Node

As you already know Altamira includes 158 main compute nodes where all the executions must be done. For security reasons, it is not possible the connection directly to the worker nodes and all the executions must be allocated in the nodes through the batch system queue (see below how to submit jobs).

Running Jobs

As is defined above SLURM is the utility used at Altamira for batch processing support, so all jobs must be run through it. This part provides information for getting started with job execution at Altamira, (see the official slurm documentation to get more information on how to create a job) .

NOTE: In order to keep the login nodes in a proper load, a 10 minutes limitation in the CPU time is set for processes running interactively in these nodes. Any execution taking more than this limit should be carried out through the queue system.

Manage Jobs

A job is the execution unit for the SLURM. A job is defined by a script containing a set of directives describing the job, and the commands to execute.

These are the basic directives to submit jobs:

- **sbatch <job script>:** submits a *job script* to the queue system (see below for job script directives).
- **squeue [-u user]:** shows all the jobs submitted by all users or by the user if you specify the **-u** option.
- **scancel <job id>:** removes his/her job from the queue system, canceling the execution of the job if it was already running.
- **scrontab [-u user] <file>**: set, edit, and remove a user's Slurm-managed crontab. This file can define a number of recurring batch jobs to run on a scheduled interval. Use the same sintaxis as cron. More info at scrontab.

On the other way, you can also launch your jobs in an interactive way to run your session in one of the compute node you have requested (used eventually for graphical applications, etc) :

```
srun -N 2 --ntasks-per-node=8 --pty bash
```

this requests 2 nodes (-N 2) and we are saying we are going to launch a maximum of 8 tasks per node (--ntasks-per-node=8). We are saying that you want to run a login shell (bash) on the compute nodes. The option --pty is important. This gives a login prompt and a session that looks very much like a normal interactive session but it is on one of the compute nodes.

Job directives

A job must contain a series of directives to inform the batch system about the characteristics of the job and you can configure them to fit your needs. These directives appear as comments in the job script and usually at the top just after the shebang line, with the following syntax:

```
#SBATCH option=value
```

Note that these directives:

- start with the #SBATCH prefix
- are always lowercase
- have no spaces in between.
- don't expand shell variables (they are just shell comments)

This table describes the common directives you can define in your job (see below an example):

Directive	Description	Default value
`--job-name=value`	The name of the job that appears in the batch queue	*script_name*
`--partition=...`	The name of the queue in slurm	*compute*
`--output=...`	The name of the file to collect the standard output of the job. The %j part in the job directives will be sustitute by the job ID.	*file-%j.out*
`--error=...`	The name of the file to collect the standard error of the job. The %j part in the job directives will be sustitute by the job ID.	*file-%j.err*
`--chdir=...`	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.	submitting directory
`--qos=...`	Quality of Service (or queue) where the job is allocated. By default, a queue is assigned for the user so this variable is not mandatory.	main
`--time=...`	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. The format can be: m, m:s, h:m:s, d-h, d-h:m or d-h:m:s	qos default time limit
`--ntasks=...`	The number of processes to allocate as parallel tasks.	1
`--cpus-per-task=...`	Number of processors for each task. Without this option, the controller will just try to allocate one processor per task. The number of cpus per task must be between 1 and 16, since each node has 16 cores (one for each thread).	1
`--ntasks-per-node`	The number of tasks allocated in each node. When an application uses more than 3.8 GB of memory per process, it is not possible to have 16 processes in the same node and its 64GB of memory. It can be combined with the cpus_per_task to allocate the nodes exclusively, i.e. to allocate 2, processes per node, set both directives to 2. The number of tasks per node must be between 1 and 16.	1

`--mem-per-cpu`	Minimum memory required per allocated CPU. Default units are megabytes unless the SchedulerParameters configuration parameter includes the "default_gbytes" option for gigabytes.	DefMemPerCPU

Job environment variables

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

Variable	Description
SLURM_JOBID	Specifies the job ID of the executing job
SLURM_NPROCS	Specifies the total number of processes in the job. Same as **-n, --ntasks**
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
SLURM_NODELIST	Specifies the list of nodes on which the job is actually running
SLURM_SUBMIT_DIR	The directory from which **sbatch** was invoked.
SLURM_MEM_PER_CPU	Memory available per CPU used

Job examples

Example for a sequential job:

```
#!/bin/bash
#
#SBATCH --job-name=hello
#SBATCH --output=hello.out
#SBATCH --ntasks=1
#SBATCH --time=10:00
# From here the job starts
# To clean and load modules defined at the compile and link phases
module purge
module load ...

# echo of commands
set -x

# To compute in the submission directory
cd ${SLURM_SUBMIT_DIR}

# execution
hostname
python example.py
```

Example for a parallel job - MPI Parallel job.

Take into account that srun only works with the version 4.1.0 of OPENMPI. Other versions prompts an error. If you use other version sof OPENMPI use **mpirun** by default when the program is called.

```
#!/bin/bash
#
#SBATCH --job-name=hello
#SBATCH --output=hello.out
#SBATCH --ntasks=1
#SBATCH --cpus-per-task= 4 # The job spawns in 4 cores
#SBATCH --time=10:00
# From here the job starts
# srun only works with th
module load OPENMPI/4.1.0

# echo of commands
set -x

# To compute in the submission directory
cd ${SLURM_SUBMIT_DIR}

# number of OpenMP threads
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

# Binding OpenMP threads on core
export OMP_PLACES=cores

# execution with 'OMP_NUM_THREADS' OpenMP threads
srun openmp.sh
srun sleep 60
```

Examples of scrontab files:

To create a job that would run at the beginning of each hour, uc00 account and have a walltime of 1 minute, you would add the following to **scrontab.**

```
DIR=/home/user1
#SCRON -p high
#SCRON -A uc00
#SCRON -t 1:00

@hourly $DIR/multi-job.sh
```

To have a job run every Wednesday, every other hour during the work day, each of the first five minutes of the hour and again at the thirty minute mark, you would add the following to scrontab.

```
1-5,30 8-17/2 * * wed $DIR/multi-job.sh
```

Software

Modules Enviroment

The Environment Modules package provides for the 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 and atomically, 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.

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

- **module list** shows all the modules you have loaded
- **module avail** shows all the modules that user is able to load
- **module load** let user load the necessary environment variables for the selected modulefile (PATH, MANPATH, LD_LIBRARY_PATH...etc)
- **module unload** removes all environment changes made by module load command
- **module switch** acts as module unload and module load command at same time

Job submitting with Modules

We need to do the loading of the needed applications inside the job script to load them in the worker nodes once the jobs require them.

Acknowledgment in publications

Add acknowledge to IFCA at the University of Cantabria for the use of Altamira supercomputer with a text similar to:

'We acknowledge Santander Supercomputacion support group at the University of Cantabria who provided access to the supercomputer Altamira Supercomputer at the Institute of Physics of Cantabria (IFCA-CSIC), member of the Spanish Supercomputing Network, for performing simulations/analyses.'

Getting Help

IFCA provides to users consulting assistance. User support consultants are available during normal business hours, Monday to Friday, 09 a.m. to 17 p.m. (CEST time).

User questions and support are handled at:

- Spanish Supercompiting Network (RES) at <res_support@ifca.unican.es>
- UC groups or *Servicio Santander de Supercomputación* users at <ssc@unican.es> or <res_support@ifca.unican.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 or log files.