Getting Started Guide

1. Introduction

This document gives some information that you will need during the course hands-on labs. Please, read it carefully.

2. MinoTauro GPU cluster

2.1. Change your password

Use an SSH client program to login to **dt01.bsc.es** using the training account login information provided to you. **This machine is only used for password setup**, the machines you will have to connect during the hands on labs of the course are detailed below. Once logged in, follow the steps that appear in the terminal to change your default password for a new one of your choice (If no steps appear after you login, execute the command *passwd).* Note that there could be some restrictions to ensure that passwords have a minimum strength level (a minimum number of characters, use of numbers and/or punctuation symbols, etc.). Once you finish changing you password, you can logout from this machine and connect to the cluster login node.

Note: the password change may take ‘a couple’ of minutes to be effective.

2.2. Cluster usage

With your SSH client login to **mt1.bsc.es** using the provided username and your new password. If you just changed your password, it may take some time (up to 10 minutes) for the change to be effective, thus it is possible that you cannot login during this time.

In this section, we present a brief description of the main commands needed to work in the cluster. For further information about the MinoTauro cluster, take a look to the MinoTauro User's Guide. This guide contains detailed information about the MinoTauro Cluster, its job management system, etc.

The MinoTauro cluster uses a job management system to control access to the computing nodes. To run the labs, you will need to use the commands provided by the system to submit jobs for execution, check their state, or cancel them if necessary:

Submitting a job

To submit a job, use the command *sbatch <job_script>,* like in the following example:

```
$ sbatch ./job.sh
Submitted batch job 170798
```

After you execute the command, you shall see a message with the ID of the newly created job (170798 in the example above). The argument of *sbatch* (<job_script>) is the path of a shell script which contains a set of directives required by the job queue system, and the command to execute the program (see the example below). Some labs already provide a job.sh file, just note that to run the different programs in a lab, you will have to edit the last line of the script to execute the corresponding program. If a lab doesn't provide a job script, make a copy of a previous one and edit it accordingly for the current lab.

```
#!/bin/bash
#SBATCH --job-name=MyJob
#SBATCH --workdir=.
#SBATCH --output=%j.out
#SBATCH --error=%j.err
#SBATCH --time=00:05:00
#SBATCH --ntasks=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1

<some command(s)>
```

Query job status

To query job status use the *squeue* command. You shall see all your submitted jobs that are waiting or running in the cluster.

A job has the PENDING state when is waiting in the queue:

```
$ squeue --long
JOBID  NAME  USER     STATE   TIME TIMELIMIT CPUS NODES NODELIST(REASON)
170798 MyJob nct00002 PENDING 0:00 10:00     6    1     (Priority)
```

A job has the RUNNING state when it is executing. When a job is in the RUNNING state, the TIME column shows the elapsed time since the job started executing, and the NODELIST (REASON) column indicates on which cluster's node/s the job is running.

```
$ squeue --long
JOBID  NAME  USER     STATE   TIME TIMELIMIT CPUS NODES NODELIST(REASON)
170798 MyJob nct00002 RUNNING 0:04 10:00     6    1     nvb118
```

When the job has finished and the job queue system has finished cleaning the job resources, the job won't appear anymore in the list:

```
$ squeue --long
JOBID NAME USER STATE TIME TIMELIMIT CPUS NODES NODELIST(REASON)
```

Canceling a job

To delete a job you can use the command *scancel <job ID>*

```
$ scancel 170798
```

When your job is finished, you can check the program output in the files “XXX.out” and “XXX.err”, where “XXX” is the job ID number. Files with “.out” extension store the execution output log (stdout). Files with “.err” extension store the error or warning messages printed during the execution (stderr).

Note: Be aware that interruptions to your SSH connection may cause you to lose unsaved work. If you have reason to believe your connection to the cluster may be unstable, you may wish to use an FTP program with the SFTP protocol and your provided login information to retrieve the source files to your local machine for editing and upload them back to the cluster to compile and execute the programs.

Execution of GUI applications

In some of the labs we may need to use GUI applications. The best way to view the windows of a GUI app running in the cluster is to use the X forwarding feature of ssh. Setting up X forwarding to be able to use remote GUI apps depends on the OS installed in your laptop;

Linux

In this case you just have to add the -X flag when connecting to the cluster through ssh:

```
$ ssh -X <username>@mt1.bsc.es
```

If any error appears while trying to run a GUI app, you may try using the -Y flag instead of -X.

Windows

Windows doesn’t handle X GUIs by default. There are several apps for Windows that enable this functionality, but many of them are not free. If you already have an X server emulation app installed in your laptop, you can use it for these labs. If not, we did some tests with the free version of `MobaXterm` and seems to work well. You can download it from:

http:/mobaxterm.mobatek.net/

Mac OS

On Mac OS, the X windows system is called XQuartz. Mac OS X 10.5, 10.6 and 10.7 installed it by default, but as of 10.8 Apple has dropped support and directs users to the open source XQuartz. You can install XQuartz from the OS distribution media or download it from https://www.xquartz.org/.

When connecting to the cluster through ssh add the -Y flag (not -X) to enable X forwarding:

```
$ ssh -Y <username>@mt1.bsc.es
```

The Labs

Use an SSH client program to login to **mt1.bsc.es** using the training account login information provided to you. Your home directory can be organized in any way you like.

Obtain the labs source code

To unpack the code for the lab assignments, execute the following command in the directory you would like the labs to be deployed:

```
$ tar -zxf ~nct00021/mv19_mpi.tgz
```

Check that a directory containing the labs has been created:

```
$ cd mv19_mpi
$ ls
```

The last command should list the lab directories and a few files.

Complete the source code

Your task during the hands-on labs is to complete the provided source code, applying the knowledge obtained during the lectures. Search for *FILLME* comments in the source code and Makefile files to know the spots where changes are required.

If you don't know/remember the parameters of MPI functions and/or their order, feel free to check any MPI API reference documentation online.

Compiling and running

To compile the labs just run

```
$ make
```

To run the program with the cluster's job scheduler use the command

```
$ sbatch job.sh
```

as explained before. Edit the last line of *job.sh* to execute the corresponding program. If the job script is not provided, copy the script from the previous lab and edit it accordingly.

The output of the *stencil_mpi_...* programs is a printed message (in the stdout file) and a bitmap image. There is a reference image in the base directory of the labs, in case you want to compare your output with it. To view the images, you have to download them (with *scp* or an *sftp* client) and view them in your laptop.