People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University Mustapha Stambouli of Mascara
Faculty of Exact Sciences

HPC Cluster Emir: User Guide

Presented by

Benaoumeur Bakhti

Supercomputing Center: University of Mascara

Mascara 2021

Contents

1 Introduction

This document is intended for anyone who is new to using the High Performance Computing (HPC) Cluster or Supercomputer and wants to work with the HPC Cluster **Emir** of the University of Mascara. It is also intended for people in other universities in Algeria as all the existing Clusters in Algeria (belonging to the CERIST) share the same architecture and all are using SLURM (see sections below) for cluster resource management. This guide can be helpful and may also contain important information for experienced users who are already familiar with the subject. This guide does not claim to be complete, but rather should, on the contrary, enable the simplest possible introduction to work with Clusters. Further extension of this guide is a work in progress and will be shared in the near future.

The cluster discussed here is the HPC-Emir computing cluster of the University of Mascara. An important thing to note here is that an HPC may not be faster than one small single computer only if cluster parallelism is exploited. The latter becomes a hot topic in modern computing and groundbreaking scientific discoveries have been made based on using the HPC. Thus, people are strongly encouraged to take part in this development. The strength of the HPC cluster lies in the large number of tasks that can be performed simultaneously. Parallelism can be done either via programming many processors (CPUs) using MPI (Message Passing Interface) or OpenMP (Open Multi-Processing) or via parallel programming of the GPU of the graphic card using either CUDA (Compute Unified Device Architecture) or OpenCL (Open Computing Language). CUDA works only with Nvidia graphic cards, however, programming with OpenCL has been implemented on a vast array of graphic cards including AMD, Intel, and Nvidia GPUs. The programming language itself is not the aim of this tutorial but the reader is advised to check the large amount of information, examples and tutorials on the different programming languages on the web.

Due to the necessity of High Performance Computing, almost all software in physics, chemistry, biology or engineering have been extended to work on HPC clusters and parallel computers. Thus, this tutorial is intended to provide the teachers/researchers and students of the University of Mascara with the necessary tools to use correctly the HPC Cluster Emir.

Finally, if you have any questions, suggestions for improvement or you find errors in the text, please do not hesitate to send me an email to benaoumeur.bakhti@univ.mascara.dz or to hpc.emir@univ-mascara.dz.

Have fun and success working with the Cluster

Mascara 2021 B. Bakhti

HPC Cluster Emir

2 HPC Custer Emir Hardware

The HPC Cluster Emir has been installed in the University of Mascara in 2015. It is manufactured by BULL, a leading company in the technology of Supercomputers. The cluster has in total 35 nodes. One node is dedicated to the administration and management of the cluster. One node is dedicated to the storage and it has a capacity of $24000GB$ ($8x3000GB$ HDD). One node is dedicated to visualization. It has a storage capacity of $2x500GB$ HDD and a graphic card of type PYNY nVidia Quadro $K5000$, $4GB$ GDDR5 and 1536 CUDA cores (see `https://www.gpuzoo.com/GPU-NVIDIA/Quadro_K5000.html` for more specifications). The remaining 32 nodes are dedicated to computation.
One each one of the 35 nodes, there are two Intel Xeon $E5-2620v2$ CPUs (processors). Each CPU has a RAM of $64GB$ and a frequency of $2.5-3.3GH$. So in total, the computes nodes have 640 cores.
The communication between you (the user) and the HPC cluster can be done via Ethernet. The communication between all the nodes of the cluster is done via the Infiniband interconnect with an extremely fast bandwidth ($40GB/s$).
The table below summarizes the resources of the HPC-Emir

Matériels	Configuration
1 Management Node	Bullx R425-E3 : CPU 2 x E5-2670v2 10 cores 2.5 Ghz RAM 64GB@1600MHz - HDD 500GB
1 Storage Node	Bullx R423-E3 : CPU 2 x E5-2620v2 2.1Ghz RAM 64GB@1600MHz - HDD 8 x 8000GB
1 Visualization Node	Bullx R425-E3 : CPU 2 x E5-2670v2 10 cores 2.5 Ghz RAM 64GB@1600MHz - HDD 2x500GB
32 Compute Nodes	Bullx R425-E3 : CPU 2 x E5-2670v2 10 cores 2.5 Ghz RAM 64GB@1600MHz, 500GB

3 Accessing the HPC Cluster Emir

Access to the Cluster Emir is available via encrypted and secured connection **ssh**. The ssh program allows the user to open a text console session on a remote computer. When a user connects to a computer at University, the user must enter their **username** and the appropriate **password**. This is done

from a shell or terminal prompt on a Linux or Mac OS system. On a windows machine, the user can access the cluster using a graphical ssh program under Windows, such as PuTTY (Cygwin or MKS can also be used).
From inside the University, you can access your account as follow:

3.1 On Linux

Connections to the Cluster via ssh can be made from a terminal. Once the terminal window is open, the syntax of the commands is usually on all Linux systems (Ubuntu, RedHat, Fedora, CentOS, Debian, SUSE,...). On terminal type (see Fig. (2))

ssh username@172.16.71.241

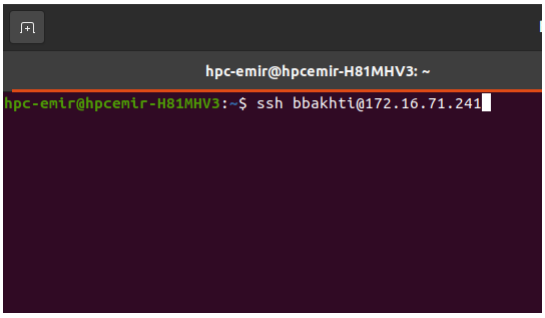

Figure 1

then press Return (Enter) on the keyboard. These commands should bring up another line asking you to type in your password (note that nothing is shown on the screen when you type the password).

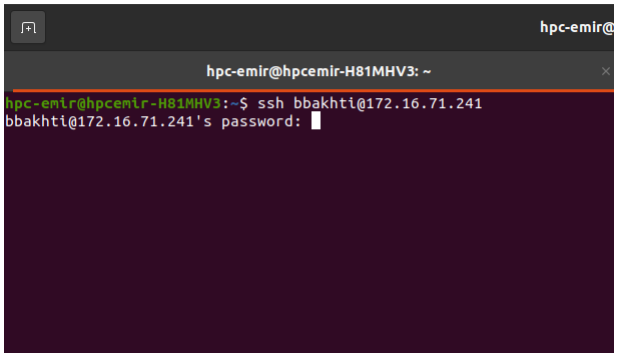

Figure 2

Enter your password, and press Return (see Fig. (3)). username must be replaced by your account name. For example, my user name is **bbakhti**, login to the cluster can be done as (see Fig. (2)):

ssh bbakhti@172.16.71.241

You are now logged in on the HPC Cluster (see Fig. (4)), and can use any of the Linux, module or queue system commands described in this manual.

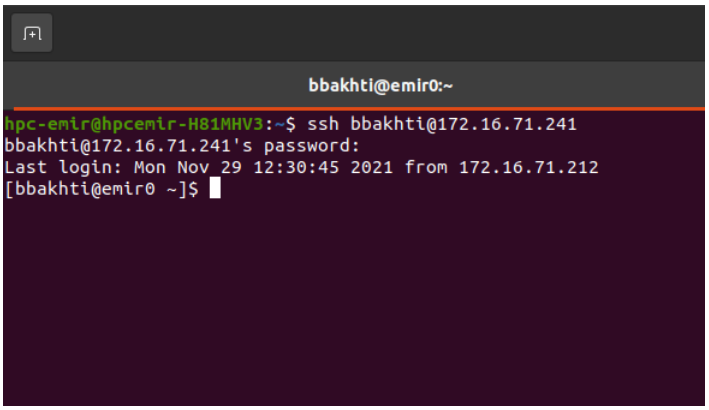

Figure 3

3.2 On Windows

Windows does not come with a ssh program, but several free ssh programs are available. The easiest to use, free ssh program is **PuTTY**. Here you need to

install putty on your windows computer. Depending on your windows version, you can download Putty from `https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html` (see Fig. (5)).

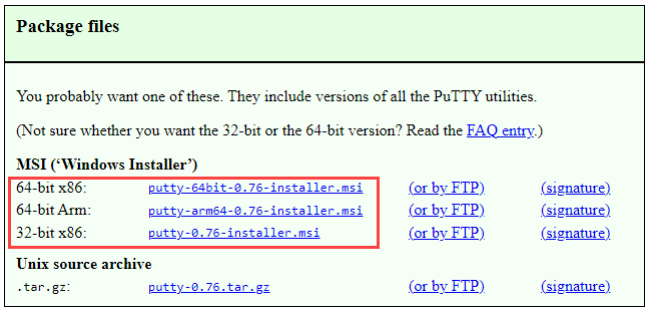

Figure 4

The installation can be done following the steps below:
1- Double click on the downloaded file.
2- Click **Next** on the welcome screen (see Fig. (6)).

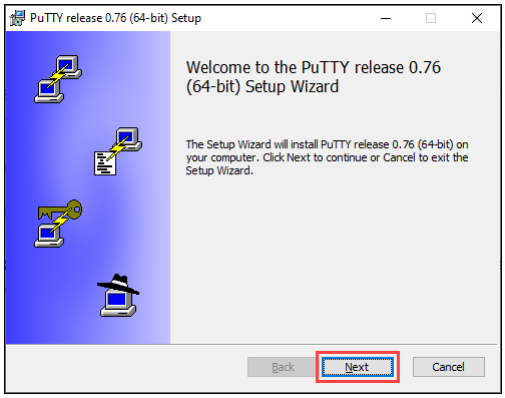

Figure 5

3. Click **Next** if you don't need to modify the installation path (see Fig. (7)). Click **Change**... to specify another path (but this is not necessary).

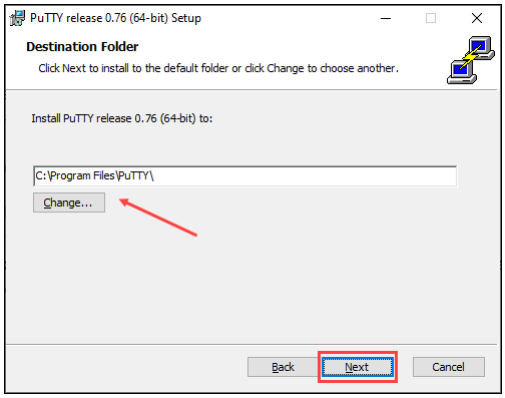

Figure 6

4- Click **Install**(see Fig. (8)).

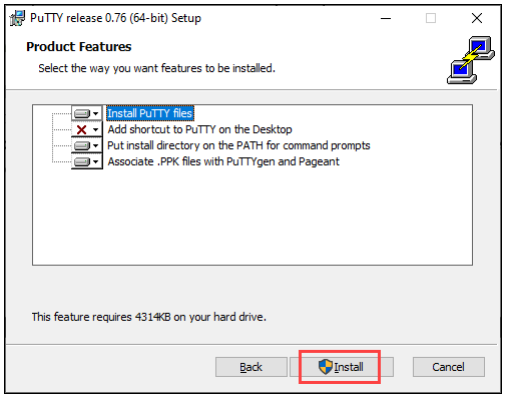

Figure 7

5. Upon completing the installation, the program shows a 'Setup complete' screen. Check/uncheck the View README file option if you want to see the developer's notes. Click **Finish** to exit the installer (see Fig. (9)).

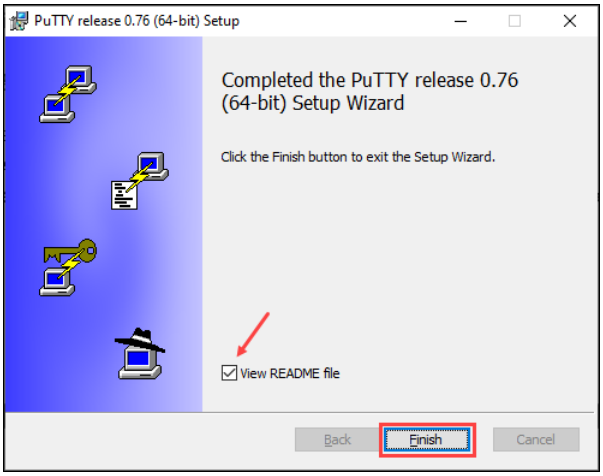

Figure 8

Putty is installed now on your PC. To connect to the Cluster with PuTTY, first, double click on the PuTTY icon on the Windows desktop or Laptop or search PuTTY in the search bar and press Enter (see Fig. (10)). This will open the PuTTY Configuration window.

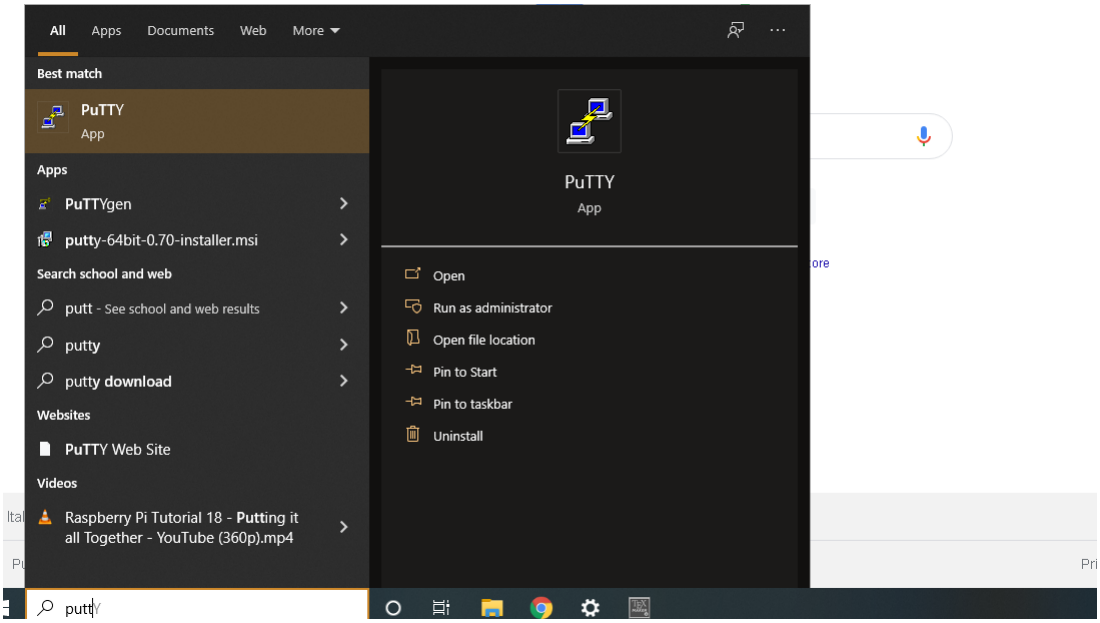

Figure 9

In the PuTTY **configuration window** (see Fig. (11)), choose the **ssh** for the connection type, keep the port 22 and type in the **hostname** or **IP address** of the cluster.

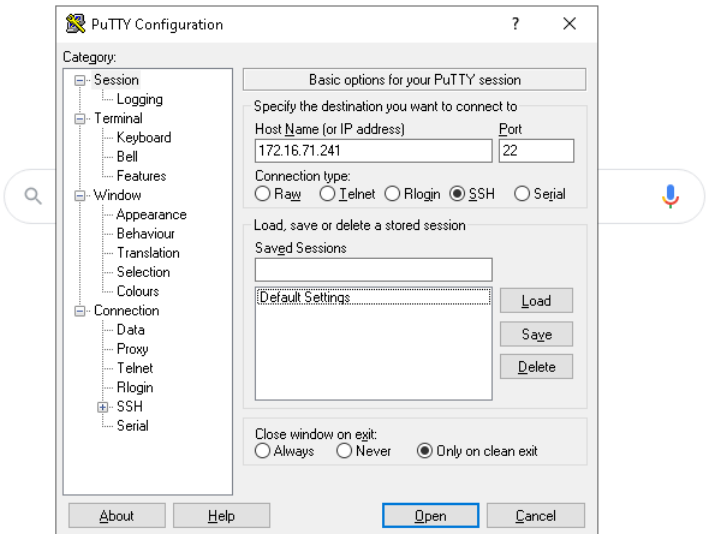

Figure 10

Windows may pop up a security warning message that requires you to click on "yes", or "Allow", or "Run", or "Continue" in order to allow the PuTTY software to run Fig. (12)). Click **yes**.

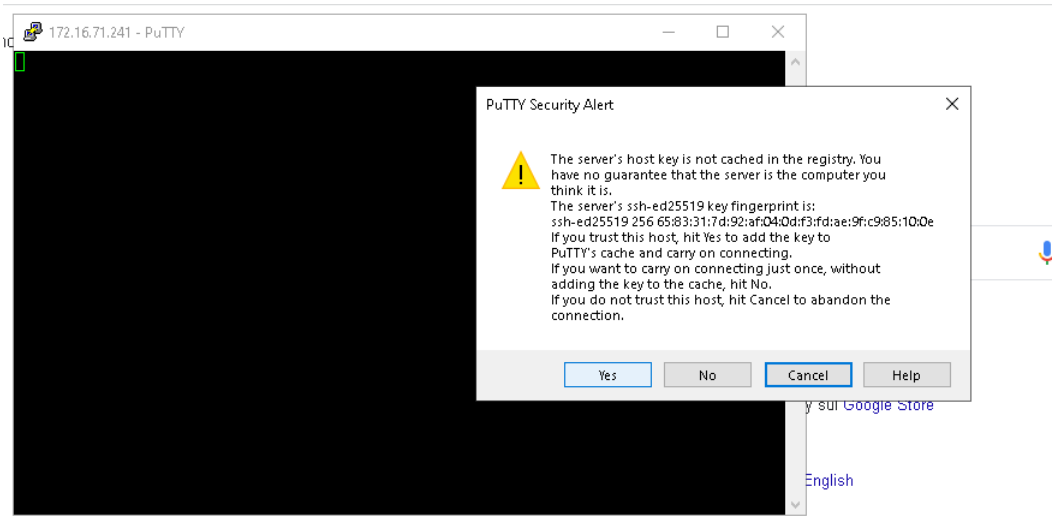

Figure 11

The PuTTY terminal window will appear with the "**login as:**" prompt Fig. (13)) . Enter your **username**, and press Return.

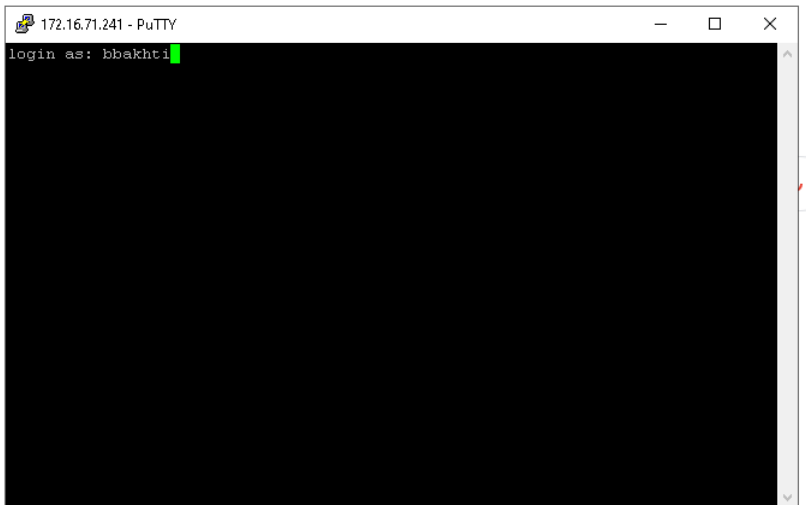

Figure 12

Next, the "**Password:**" prompt will appear Fig. (14)). Enter your pass-

word and press Return. Note that nothing is shown on the screen when you type the password,

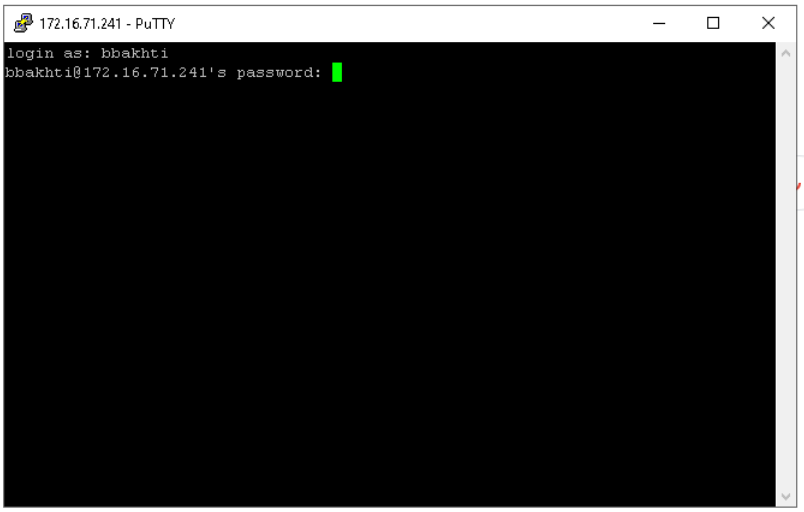

Figure 13

Congratulation! you should be in the terminal console of your account Fig. (15)).

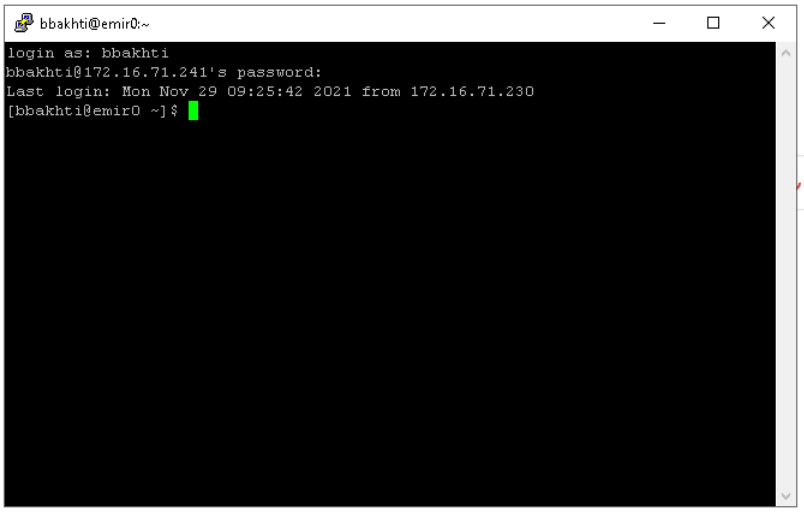

Figure 14

If you want to exit your account (**log off** the Cluster), simply type **exit** and press Return.

3.3 Transferring files between PC and Cluster

Files can be transferred between your PC on the Cluster using the **scp** command. To transfer the file to the cluster, open the terminal prompt on Linux or PuTTy on Windows. Then you should navigate to the path where your file is located. Suppose your file is a python file named $fileExample.py$ (but it can be a file with any extension such as c, cpp, doc, pdf,...). To transfer the $fileExample.py$ to a folder named $myFolder$ in your cluster home directory, use the following command

scp fileExample.py / username@172.16.71.241:./myFolder

Then press Return. The file should be transferred. An example is shown in Fig. (16)

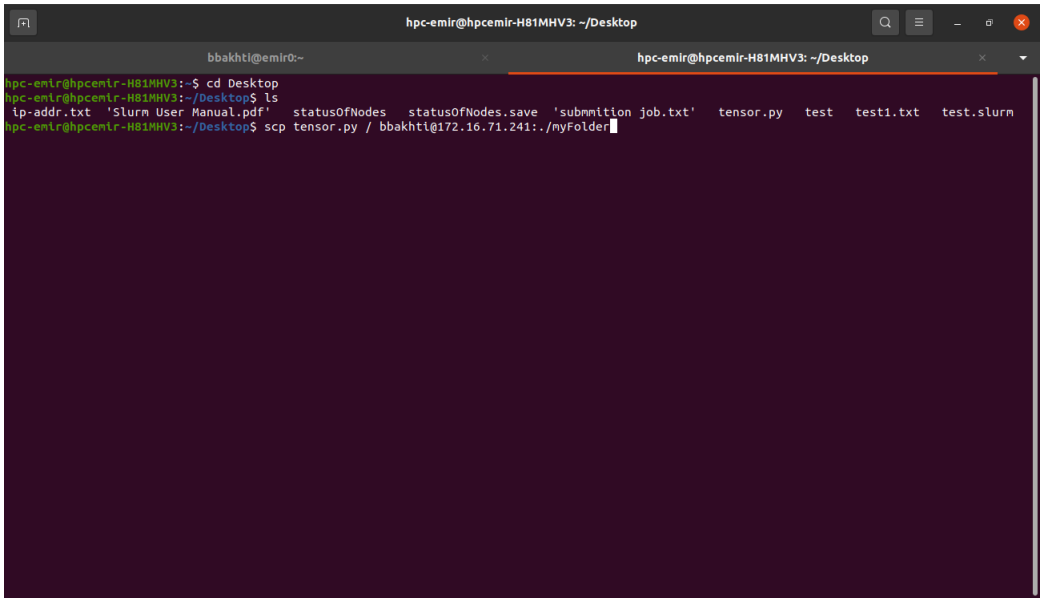

Figure 15

In this example; the file **tensor.py** will be transfered from the Desktop to myFolder in the cluster home directory.

To move a folder named *folderExample* to the *myFolder* in the cluster use.
scp -r folderExample / username@172.16.71.241:./myFolder

and press Return.
Transfer of files (for example output file) from the Cluster to your computer can be done as follow. Open a terminal prompt on your computer. Navigate to the place where you want to move the file to. Suppose the file you want to move is named *output.txt* and it is exists in the *myFolder* folder in your home directory in the cluster. use the following command to move the file to your PC

scp usename@172.16-71.241:./myFolder/output.txt .

and press Return. Note the space between *output.txt* and the dot is mandatory. Example is shown in Fig. (17)

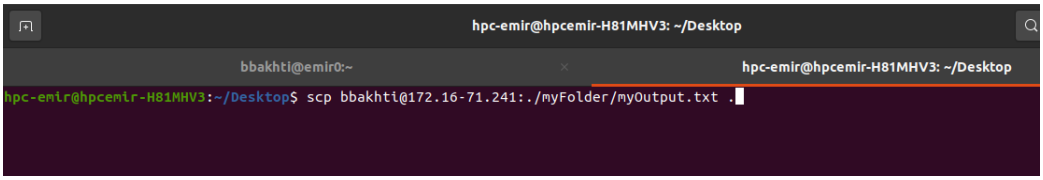

Figure 16

In this example; the file **myOutput.txt** will be transferred from the myFolder folder in the cluster home directory to the Desktop of my PC:

3.4 Submitting Job to the Cluster

Executing a job on the HPC Cluster Emir will be done using the following steps:

- Determine the resources necessary for the specific job (number of nodes, number of CPUs, required time, GPU, amount of memory,...).
- Create a Batch job script.
- Submit the job to the scheduler.
- Check the job status.

- Retrieve job information and output.

All these steps will be performed using a specific software called **SLURM**. Detailed on these operations will be given below and for more information on Slurm, please check `https://schedmd.com/` or `https://hpc.llnl.gov/training/tutorials#trainingmaterials`.

3.4.1 What is Slurm

Slurm (**S**imple **L**inux **U**tility for **R**esource **M**anagement) is an open source, fault-tolerant, and highly scalable cluster management software. It is the most commonly used job scheduler for Linux-based HPC clusters and is the software we use to run all jobs on the HPC Cluster Emir. Slurm performs several important functions: it allows the user to allocate time and resources on the compute node(s) to perform a job. It also allows the user to start and monitor the status of the job as it runs on the worker nodes. Finally, it queues and balances job submissions for all users on the cluster.

3.4.2 Determine resources for job

Choosing appropriate resources for your jobs is essential to ensuring your jobs will be scheduled as quickly as possible while wasting as little resources as possible. There are no recipes to follow when choosing the cluster resources for your job. But generally, you will learn this with experience working with the cluster.
The key resources you need to optimize for are:

- Time allocation
- Nodes and Cores allocation
- Memory-allocation

Time requirements are highly dependent on how many CPU cores your job is using - using more cores may significantly decrease the amount of time the job spends running.
In order to determine your CPU requirements, you should investigate if your program/job supports parallel execution. If the program only supports serial processing, then you can only use 1 CPU. If your job/program supports multiple cores, you need to allocate multiple CPUs. At the moment, the

Maximum number of cores that you can allocate is 30 cores. For a GPU job, you can only use the visualization node of the HPC-Emir Cluster.
For the memory allocation, submit your job **sbatch** by specifying very generous memory and time requirements to ensure that it runs to completion and also using the $--$**mail**$-$**user=** and $--$**mail**$-$**type=ALL** parameters to receive an email-report. The mail message will list the maximum memory usage (maxvmem/MaxVMSize) as well as the wallclock time used by the job.
Remember, the larger your request, the longer it will take for the resources to become available and the time taken to queue is highly dependent on other cluster jobs.

3.4.3 Create a Batch script for the job

I present here some slurm batch scripts samples that can be used as a template for building your own Slurm submission scripts for use on HPC-Emir. Please make sure you understand each #SBATCH directive before using the script to submit your jobs. First, you need to create an empty batch script (with **.sh** or **.slurm** extension). On both Linux (terminal) or Windows (PuTTy console) you can use either the command **nano** or the command **vi** as follow (see Fig .(18) forWindows):

nano batch_example.sh

vi batch_example.sh

Both commands bring a a similar empty file.

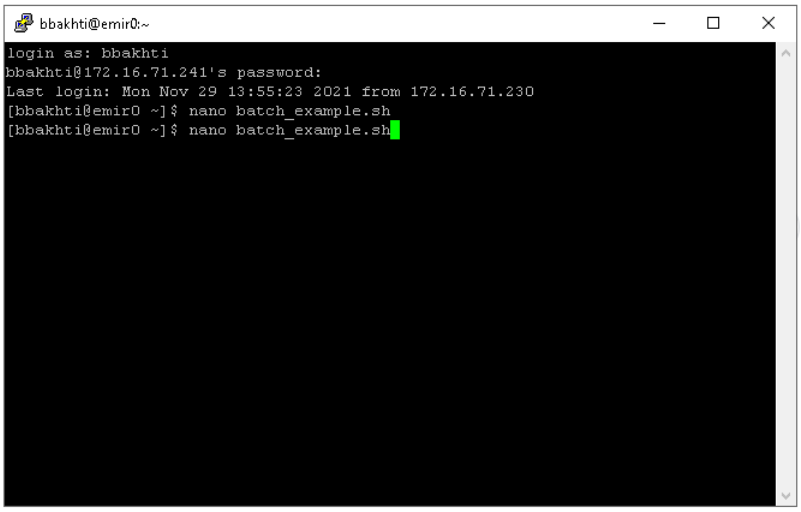

Figure 17

After pressing enter you should get an empty file like the one displayed in Fig .(19) (for Linux terminal, but you get exactly the same thing for PuTTy console)

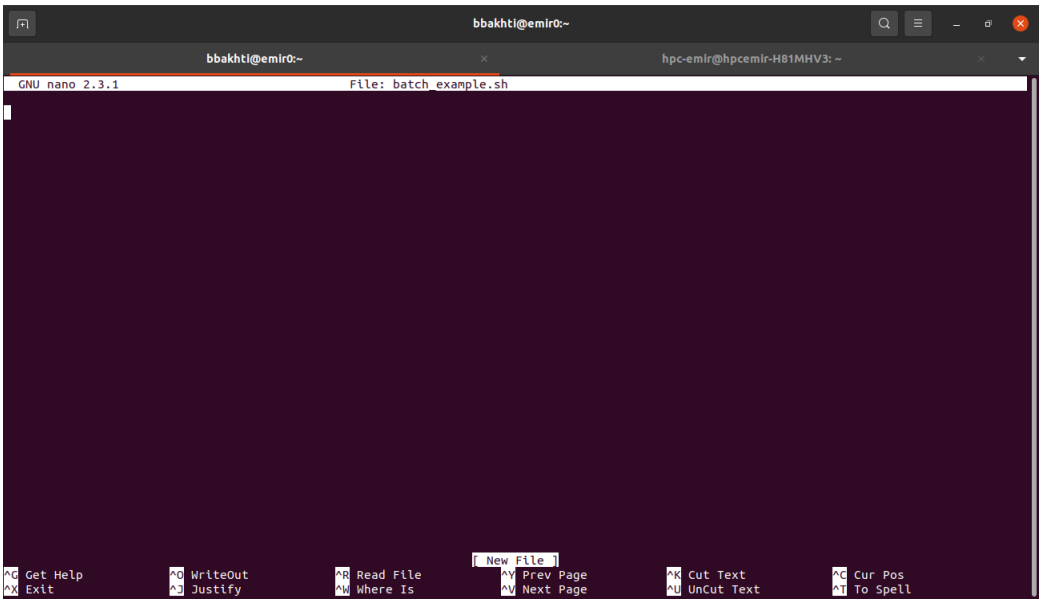

Figure 18

Copy the following batch file into the empty terminal screen (see Fig .(20) for both Windows putty and Linux terminal). If you have used **vi** editor to create the batch file, then you need to press the letter "**i**" to insert or modify text.

#!/bin/bash
 # set name of the job
#SBATCH −job−name=mpi_program
#SBATCH −−partition=all
 # set the number of nodes (2 in the present case)
#SBATCH −−nodes=2
 # number of processes per node (processors cores)
#SBATCH −−ntasks−per−node=20
 # Job memory request #SBATCH −−mem=1gb # set max wallclock time (D-hh:mm:ss), 40 min in the present case
#SBATCH −−time=0-00:40:00
 # load the required modules
module load openmpi
 # Define the mail address for notifications
#SBATCH −−mail−type=end
 # When to send mail notifications Options: BEGIN,END,FAIL,ALL
#SBATCH −mail-user=email@univ-mascara.dz
#SBATCH −−error=mpi_program.err
#SBATCH −−output=mpi_program.out
srun ./mpi_program

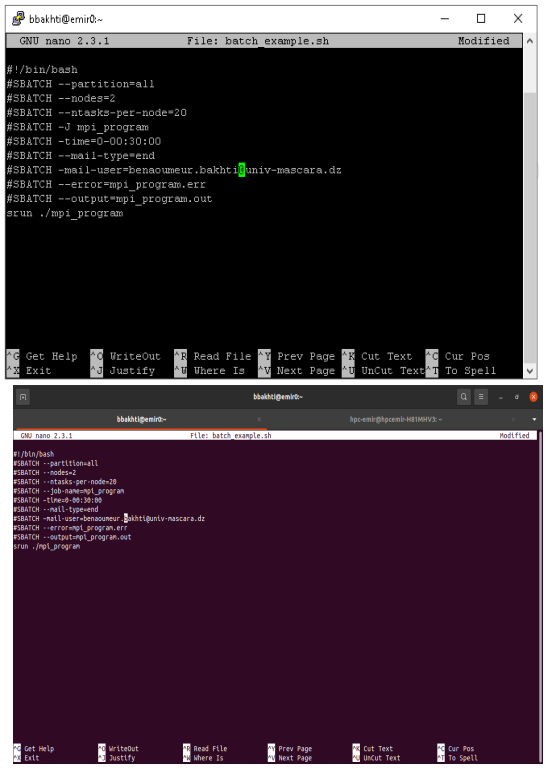

Figure 19

To exit and save the batch file in the **nano** editor, press **Ctlr+X** then **y** then **Return**. To exit and save the batch file in the **vi** editor, press first **Esc** in the keyboard, then type **:wq**, then press **Return** and you are done. Here are some examples of SLURM script for running a batch job on the HPC cluster Emir with **MPI-C/C++**, **MPI-Fortran**, and **MPI-Python**:

MPI-C++ Batch file

#!/bin/bash

#SBATCH −p all
#SBATCH −J MPIProgram
#SBATCH −o MPIProgram-%j.out
#SBATCH −e MPIProgram-%j.err
 # Run for a maximum time of 0 days, 12 hours, 00 mins, 00 secs

```
#SBATCH -t 0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20
#SBATCH --mem-per-cpu=4000
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<your email>

# list modules loaded by default.
module list
# prints current working directory
pwd
# prints the date and time
date

# run the MPI job
mpirun my_program.cpp
```

MPI-Fortran Batch file

```
#!/bin/bash

#SBATCH -p all
#SBATCH -J MPIProgram
#SBATCH -o MPIProgram-%j.out
#SBATCH -e MPIProgram-%j.err
#SBATCH -t 0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20
#SBATCH --mem-per-cpu=4000
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<your_email>
# list modules loaded by default(GNU8 compilers and OpenMPI3 MPI libraries)
module list
# swap the MPI library from the default openmpi3 to mpich.
module swap openmpi3 mpich
module list
pwd
date
```

mpirun my_program.f

MPI-Python Batch file

#!/bin/bash
#SBATCH −p all
#SBATCH −J MPIProgram
#SBATCH −o MPIProgram−%j.out
#SBATCH −e MPIProgram−%j.err
#SBATCH −t 0-12:00:00
#SBATCH −−nodes=1
#SBATCH −−ntasks−per-node=20
#SBATCH −−mem−per−cpu=4000
#SBATCH −−mail−type=ALL
#SBATCH −−mail−user=<your_email>
module list
pwd
date
mpirun python my_program.py

3.4.4 Submit and check the status of a job

There are 2 commands for job allocation: **sbatch** is used for batch jobs and **salloc** is used to allocate resources for interactive jobs. The format of these commands:

- sbatch [options] jobscript [args...]
- salloc [options] [¡command¿ [command args]]

If the batch file is named "*my_batch.sh*" or "*my_batch.slurm*", then you submit your job using

sbatch my_batch.sh
or
sbatch my_batch.slurm
This command will automatically queue your job using SLURM and produce

a job ID number (shown below). You can check the status of your job at any time with the ***squeue -j <JOB_ID>*** command.

squeue -j job_ID

You can also stop your job at any time with the **scancel** command.

scancel job_ID

Other useful commands to check the job status is summarized in the following table

List all current jobs for a user	squeue −u username
List all running jobs for a user	squeue −u username −t RUNNING
List all pending jobs for a user	squeue −u username −t PENDING
List all current jobs in the general partition for a user	squeue −u username −p general
List detailed information for a job (useful for troubleshooting)	scontrol show jobid −dd jobid
To cancel one job	scancel jobid
To cancel all the jobs for a user	scancel −u username
To cancel all the pending jobs for a user	scancel −t PENDING −u username
To cancel one or more jobs by name	scancel –name JobName
To pause a particular job	scontrol hold jobid
To resume a particular job	scontrol resume jobid
To requeue (cancel and rerun) a particular job	scontrol requeue jobid

When running the **squeue** command, you get information (in columns) displaying the Job_IDs, names of the job, STs, the users, and the nodes. The ST column gives the state of the job, with the following codes:

- R for Running
- PD for PenDing
- TO for TimedOut
- PR for PReempted

- S for Suspended
- CD for CompleteD
- CA for CAncelled
- F for FAILED
- NF for jobs terminated due to Node Failure

3.4.5 Retrieve output

As stated before, different commands can be used to retrieve the results and output of the programs from the HPC cluster. The most used one is the **scp** command. The latter can be used as follow

scp usename@172.16-71.241:./myFolder/output.txt .

and transfer a directory or a folder use

scp -r usename@172.16-71.241:./myFolder/output.txt .

I recall that the space between *output.txt* and the dot is mandatory. Another command that can be used is the **sftp**.

3.4.6 Slurm most used Commands

- **salloc** to request interactive jobs/allocations
- **sattach** to attach standard input, output, and error plus signal capabilities to a currently running job or job step
- **sbatch** to submit a batch script (which can be a bash, Perl, or Python script)
- **scancel** to cancel a pending or running job or job step
- **sbcast** to transfer a file to all nodes allocated for a job
- **sgather** to transfer a file from all allocated nodes to the currently active job. This command can be used only inside a job script

- **scontrol** provides also some functionality for the users to manage jobs or queries and get some information about the system configuration
- **sinfo** to retrieve information about the partitions, reservations, and node states
- **smap** graphically shows the state of the partitions and nodes using a curses interface.
- **sprio** can be used to query job priorities
- **squeue** to query the list of pending and running jobs
- **srun** to initiate job steps mainly within a job or start interactive jobs. A job can contain multiple job steps executing sequentially or in parallel on independent or shared nodes within the job's node allocation
- **sshare** to retrieve fair-share information for each user
- **sstat** to query status information about a running job
- **sview** is a graphical user interface to get state information for jobs, partitions, and nodes
- **sacct** to retrieve accounting information about jobs and job steps in Slurm's database
- **sacctmgr** allows also the users to query some information about their accounts and other accounting information in Slurm's database.

4 A short introduction to Linux command

In this section, I present the list of the most used basic Linux commands:

sudo: Short for (short fir SuperUser Do), the sudo command is used when you want to perform tasks that require administrative or root permissions.
pwd: the pwd (short present working directory) command is used to find out the path of the current working directory (folder) you're in.
cd: the cd (short for change directory) command is used to navigate through the Linux files and directories. There are some shortcuts to help you navigate quickly:

- **cd ..** (with two dots) to move one directory up
- **cd** to go straight to the home folder
- **cd-** (with a hyphen) to move to your previous directory
- **cd** or **cd** $\sim$ to navigate to your home directory
- **cd /** to navigate into the root directory

ls: the ls command is used to view the contents of a directory

- **ls -R** will list all the files in the sub-directories as well
- **ls -a** will show the hidden files
- **ls -al** will list the files and directories with detailed information like the permissions, size, owner, etc.

cp: the cp command is used to copy files from the current directory to a different directory.
mv: the mv command is used to move files, although it can also be used to rename files.
mkdir: the mkdir command is used to make (create) a new directory.
rmdir: the command rmdir is used to delete a directory.
rm: the rm command is used to delete files. If you want to delete the directory (as an alternative to rmdir) use rm -r.
cat: cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output (sdout). As an example:
cat file.txt
will display the content of the text file file.txt in the terminal. You can use cat also as:

- **cat > filename**: creates a new file
- **cat filename1 filename2 > filename3**: joins two files (1 and 2) and stores the output of them in a new file (3)

touch: the touch command is used to create a blank new file through the Linux command line.
clear: the clear command is used to clear the Linux window terminal.
grep: the grep command is used you search through all the text in a given file.
kill: used to terminate manually a process.
ping is used to to check your connectivity status to a server.
top:the top command will display a list of running processes and how much CPU each process uses.
zip, **unzip**: these commands are used to compress or decompress a file.
hostname: used to display the hostname of your host/network. Adding a $-i$ to the end will display the IP address of your network.
passwd: this command is used to change password.