HPC Ecosystems Interaction
SLURM - Simple Linux Utility for Resource Management

GUANE
HPC – Basic Architecture

Diagram showing the basic architecture of a high performance computing (HPC) system, including Users, Terminal, Firewall, High Speed Network, Switch, Master Node, I/O Server, Compute Nodes, Storage, Fiber Channel, and Ethernet Network.
HPC - Base System

- NTP
- NFS
- Infiniband support
- Memory usage limits
- HPC Modules – LMOD
- PowerShell
- NHC
SLURM & MUNGE

slurm.conf

SchedulerType -> SchedulerType=sched/backfill
SelectType ->
SelectTypeParameters=CR_Core,CR_Core_Default_Dist_Block
SelectTypeParameters - >
SelectTypeParameters=CR_Core,CR_Core_Default_Dist_Block
PriorityType - > PriorityType=priority/multifactor

SLURM - NHC -> HealthCheckProgram
• Good practice in implementing LDAP is using dynamic groups that allow you to assign different levels of access to different storage spaces within the HPC platform.

• Storage spaces such as the user’s home folder and project and research group folders must have restrictions implemented through disk quotas in conjunction with LDAP.
GUANE - GpUs Advanced computing Environment

HPC – SC3UIS

Technical specifications - GUANE

16 nodes ProLiant SL390s G7
• 8 nodes:
  • 2 Intel(R) Xeon(R) CPU E5645 @ 2.40GHz.
  • 104 GB RAM
  • 1 disk SAS de 200GB
  • 8 GPU Tesla M2075
• 3 nodes:
  • 2 Intel(R) Xeon(R) CPU E5645 @ 2.40GHz.
  • 104 GB RAM
  • 1 disk SAS de 200GB
  • 8 GPU Tesla S2050
• 5 nodes:
  • 2 Intel(R) Xeon(R) CPU E5640 @ 2.67GHz
  • 104 GB RAM
  • 1 disk SAS de 200GB
  • 8 GPU Tesla S2050

Network
• 10Gbit/s Ethernet – Administration
• 40 Gb/sec Infiniband
GUANE - GpUs Advanced computiNg Environment

HPC – SC3UIS

OTHER NODES

THOR
Technical Specifications

• ProLiant DL580 Gen9
• 4 Intel(R) Xeon(R) CPU E7-8867 v3 @ 2.50GHz – 128 Cores
• 1320732708 kB – 1.2TB RAM

YAJÉ
Technical Specifications

• ProLiant ML350 Gen9
• 1 Intel(R) Xeon(R) CPU E5-2609 v3 @ 1.90GHz – 6 Cores
• 49031292 kB – 48GB RAM
• 1 NVIDIA GeForce GTX Titan X 12 GB

FELIX (Framework to Enhance artificial Intelligence applications eXecution)
Technical Specifications

• ProLiant DL580 G7
• 4 Intel(R) Xeon(R) CPU X7560 @ 2.27GHz – 64 Cores
• 131844368 kB – 128GB RAM
• 2 NVIDIA GeForce GTX Titan X 12 GB
What is SLURM?

SLURM is open-source Linux cluster management and job management software.

- Collects the resources requested by users.
- Assign priority to jobs.
- Runs the jobs on the assigned compute nodes.
- Monitoring cluster resources.
Computing nodes are grouped into logical sets called partitions that depend on their hardware characteristics or function:

**PARTITION**

- **normal***
- **guane_16_cores**
- **guane_24_cores**
- **Viz**
- **deepL**

**guane01 – guane16**

- **yaje**
- **felix**
CONNECTION TO GUANE

SSH username@ip-address or hostname

ssh user_name@toctoc.sc3.uis.edu.co

ssh guane
Use of SLURM

- Shows the information of the nodes and partitions.
- An asterisk (*) after the partition name indicates that it is the default partition.
- An asterisk (*) after the node status indicates that it is not responding.

```bash
[user_name@guane ~]$ sinfo
```

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal*</td>
<td>up</td>
<td>infinite</td>
<td>4</td>
<td>mix</td>
<td>guane[03,05,09,16]</td>
</tr>
<tr>
<td>normal*</td>
<td>up</td>
<td>infinite</td>
<td>8</td>
<td>alloc</td>
<td>guane[01-02,04,10,12-15]</td>
</tr>
<tr>
<td>normal*</td>
<td>up</td>
<td>infinite</td>
<td>2</td>
<td>idle</td>
<td>guane[06,11]</td>
</tr>
<tr>
<td>guane_16_cores</td>
<td>up</td>
<td>infinite</td>
<td>2</td>
<td>mix</td>
<td>guane[03,05]</td>
</tr>
<tr>
<td>guane_10_cores</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>guane[06]</td>
</tr>
<tr>
<td>guane_24_cores</td>
<td>up</td>
<td>infinite</td>
<td>2</td>
<td>mix</td>
<td>guane[09,16]</td>
</tr>
<tr>
<td>guane_24_cores</td>
<td>up</td>
<td>infinite</td>
<td>0</td>
<td>alloc</td>
<td>guane[01-02,04,10,12-15]</td>
</tr>
<tr>
<td>Viz</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>yaje</td>
</tr>
<tr>
<td>deepL</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>alloc</td>
<td>felix</td>
</tr>
</tbody>
</table>
Use of SLURM

```sh
squeue -u student_30
```

- Displays the job queue for user **student_30**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODE</th>
<th>Nodelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>18276</td>
<td>deepL_mafft_09_mpi</td>
<td>druedap</td>
<td>R</td>
<td>7:46:29</td>
<td>felix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18277</td>
<td>normal</td>
<td>gisaid_04</td>
<td>druedap</td>
<td>R</td>
<td>7:39:41</td>
<td>guane02</td>
<td></td>
</tr>
<tr>
<td>18282</td>
<td>guane_24_cores</td>
<td>gisaid_03</td>
<td>druedap</td>
<td>R</td>
<td>2:33:47</td>
<td>guane04</td>
<td></td>
</tr>
</tbody>
</table>

**Status**

- **R** = Running
- **PD** = Pending
- **CA** = Cancelled

```sh
[user_name@guane ~]# squeue
```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODE</th>
<th>Nodelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>17772</td>
<td>guane_24_cores</td>
<td>boinc_latorresn</td>
<td>R</td>
<td>23-08-13:22</td>
<td>guane10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18014</td>
<td>guane_24_cores</td>
<td>sim1_cbernalc</td>
<td>R</td>
<td>18-22:24:19</td>
<td>guane15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18015</td>
<td>normal</td>
<td>orcaN1_geramirez</td>
<td>R</td>
<td>18-21-48:41</td>
<td>guane01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18046</td>
<td>guane_24_cores</td>
<td>sim1_cbernalc</td>
<td>R</td>
<td>9-12-22:39</td>
<td>guane13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18252</td>
<td>normal</td>
<td>cubes3_sh_jmpachecoa</td>
<td>R</td>
<td>22-22:32</td>
<td>guane03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18275</td>
<td>guane_24_cores</td>
<td>sim1_arromero</td>
<td>R</td>
<td>8:28:06</td>
<td>guane14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18276</td>
<td>deepL_mafft_09_mpi</td>
<td>druedap</td>
<td>R</td>
<td>7:47:35</td>
<td>felix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18277</td>
<td>normal</td>
<td>gisaid_04</td>
<td>druedap</td>
<td>R</td>
<td>7:40:46</td>
<td>guane02</td>
<td></td>
</tr>
<tr>
<td>18279</td>
<td>guane_24_cores</td>
<td>sim1_arromero</td>
<td>R</td>
<td>6:09:15</td>
<td>guane12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18281</td>
<td>normal</td>
<td>bash_emvargasd</td>
<td>R</td>
<td>4:40:48</td>
<td>guane16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18282</td>
<td>guane_24_cores</td>
<td>gisaid_03</td>
<td>druedap</td>
<td>R</td>
<td>2:34:52</td>
<td>guane04</td>
<td></td>
</tr>
<tr>
<td>18283</td>
<td>guane_16_cores</td>
<td>cubes1.sh_crrcavaja</td>
<td>R</td>
<td>1:28:47</td>
<td>guane05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18284</td>
<td>guane_24_cores</td>
<td>cubes2.sh_crrcavaja</td>
<td>R</td>
<td>1:26:58</td>
<td>guane09</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use of SLURM

\texttt{srun options}

- Allows you to run an application directly with options specified by the user in \texttt{options parameters}.

[\texttt{user\_name@guane ~}]# \texttt{srun --ntasks=4 --partition=normal --label /bin/hostname}

[\texttt{user\_name@guane ~}]# \texttt{srun -n 4 -p normal -l /bin/hostname}

\begin{itemize}
  \item 2: guane01.uis.edu.co
  \item 1: guane01.uis.edu.co
  \item 0: guane01.uis.edu.co
  \item 3: guane01.uis.edu.co
\end{itemize}
Use of SLURM

**salloc** *options*

- Gets the assignment of a job with console access.
- The resources reserved for the job are those specified in *options*.
- Allows you to make an interactive reservation.

**Interactive Reservation**

```
[user_name@guane ~]# salloc --nodes=1 --partition=normal --exclusive srun --pty /bin/bash

[user_name@guane ~]# srun --nodes=1 --partition=normal --exclusive --pty /bin/bash
```
Environment Modules – Software in GUANE

- Modules are a packaging of environment variables within a script.
- One module is defined per application, which defines an appropriate environment for its execution.
- **Command list:**
  - module available
  - module load MODULE_NAME
  - module unload
  - module list
  - module purge
Use of SLURM

Environment Modules

module avail

- Shows all the modules available on the platform.

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
Environment Modules

**module load module_name**

- Loads the environment variables corresponding to the selected module *(module_name)*

```bash
[user_name@guane ~]# module load CFD/OpenFOAM/1906
```

**module list**

- List all modules that have been loaded with the *module load* command. You should keep in mind that you can load one or more modules simultaneously.
Environment Modules

**module unload** *module_name*

- Removes all environment variables corresponding to the selected module(*module_name*)

```
[user_name@guane ~]# module unload CFD/OpenFOAM/1906
```

**module purge**

- Removes all environment variables from all modules that are loaded in the current session
Use of SLURM

BATCH JOB SCRIPT

myjob.slurm

```bash
#!/bin/bash

# Resource request
#SBATCH --partition=guane_16_cores
#SBATCH --nodes=1
#SBATCH --ntask=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=1G

# Job Execution Time
#SBATCH --time=1-12:30:00

# Job name and output files
#SBATCH --job-name=myjob
#SBATCH --output=myjob.out
#SBATCH --error=myjob.err

# Loading of the environment module
module load CFD/OpenFOAM/1906

# Execution
blockMesh
```

Preliminaries

- Specify the command interpreter (Bash).
- It should always be the first line.

SLURM Directives

- They should always start with #SBATCH
  - They are ignored by bash but interpreted by SLURM.
  - Comments can be made before, between, or after directives.
  - They must be placed before loading the modules and executing the job.

Script commands

- Loading the modules required for the execution of the work
- Commands that you want to execute in the computing nodes
  - Executable of the loaded application.
  - Programming commands can be written in bash.
Use of SLURM

`sbatch batch_file`

- Sends the `batch_file` to SLURM for execution.
- If the submission is successful, SLURM returns the job ID

```
[user_name@guane ~]# sbatch myjob.slurm
```

```
[user_name@guane ~]# squeue -u druedap

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODE</th>
<th>Nodelist(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18276</td>
<td>deepL_mafft_09_mpi</td>
<td>druedap</td>
<td>R</td>
<td>7:45:29</td>
<td>1 felix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18277</td>
<td>normal</td>
<td>gisaid_04</td>
<td>druedap</td>
<td>7:39:41</td>
<td>1 guane02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18282</td>
<td>guane_24_cores</td>
<td>gisaid_03</td>
<td>druedap</td>
<td>2:33:47</td>
<td>1 guane04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Use of SLURM

```
scancel jobid
```

- Sends a signal to the job and/or its threads.
- By default, the signal sent is SIGKILL for the termination of the job.
- The job that is canceled is the one that corresponds to `jobid`.
- The `jobid` is obtained by executing the `sinfo` command.

```
[user_name@guane ~]# scancel 12345
```

- Filters can be used for job cancellation

```
[user_name@guane ~]# scancel --user=sutedent_30
```
Use of SLURM

EXAMPLE – Use of OpenFOAM in GUANE

• Download the examples *
  wget http://www.hpc.lsu.edu/training/weekly-materials/Downloads/intro_of.tar.gz

• Unzip the sample file
  tar zxf intro_of.tar.gz

• Enter the folder of the first example to test
  cd intro_of/cavity

• Create the batch job script
  nano cavity.slurm

• Launch of work in SLURM
  sbatch cavity.slurm

* Example URL:

#!/bin/bash

# Resource request
#SBATCH --partition=normal
#SBATCH --nodes=1
#SBATCH --ntasks=16
#SBATCH --ntasks-per-node=16

# Job Execution Time
#SBATCH --time=00:10:00

# Job name and output files
#SBATCH --job-name=cavity
#SBATCH --output=cavity_%j.out
#SBATCH --error=cavity_%j.err

# Loading of the environment module
module load CFD/OpenFOAM/2.4.0

# Execution
blockMesh # (generate mesh information)
iconFoam # (running the PISO solver)
foamToVTK #(convert to VTK format, optional)
Recommendations and others
Take account of

- SSH login
- Multi-factor authentication
- How Many login nodes are there?
- Connection troubleshooting guide.
- Live Status from services.
  - Nodes alive
  - Filesystems
  - Mass Storage systems
  - Planned Outages
- System Ticket platform.

- Wiki Documentation
  - Software
  - Interactive Nodes
  - Data sharing approach
  - FAQs.
• Don't run jobs in login nodes.
• Don't run many jobs at time.
• Don't use all user space on Scratch partition.
• Clean up your $HOME directory frequently

These general recommendations may change over time and may need to be adjusted for your HPC workloads

Remember that HPCs are shared systems and try avoid allocating resources which you don’t use
Job Name: grom.job

Program here:
module load gcc/5.2.0 openmpi/1.10.2 gromacs/2016.3-CUDA
srun --mpi=pmi2 --accel-bind-g --ntasks=$SLURM_NTASKS gmx mdrun -v

grom.job

### Number of nodes requested.
#SBATCH --nodes=2
### Number of tasks(processes).
#SBATCH --ntasks=4
### Amount memory per core in MB.
#SBATCH --mem-per-cpu=50000M
### Wall clock time limit in Days-HH:MM:SS
#SBATCH --time=00:05:10:00
### Send an email when the job status changes.
#SBATCH --mail-user=user@mail.com
#SBATCH --mail-type=begin,fail
### File in which to store job output.
#SBATCH --output=grom.$j.out
### File in which to store job error messages.
#SBATCH --error=grom.$j.err
### Insert code, and run your programs here ##
module load gcc/5.2.0 openmpi/1.10.2 gromacs/2016.3-CUDA
srun --mpi=pmi2 --accel-bind-g --ntasks=$SLURM_NTASKS gmx mdrun
### Submit script example: ##
### sbatch grom.job

Number of Nodes: 2
Tasks (processes): 4
Tasks per node: 2
Memory per core: 50000 MB
Walltime: 00:05:00
Output File: grom.$j.out
Error File: grom.$j.err
- Performance improvements from parallel execution do not scale linearly.
- Parallel programming allows application to take advantage of parallel hardware, serial code will not 'just work'.
- Common case of distributed memory parallelism, MPI (Message passing Interface)
Computational Performance

The graph illustrates the speedup of computational performance as the number of processors increases. The ideal, linear scaling line represents the best possible performance where doubling the number of processors doubles the speedup. Amdahl's law is shown by the blue line, which is affected by both the ideal and non-ideal sections. The green line represents efficient scaling, while the orange line indicates waste of resources, where the speedup does not increase as expected with more processors.
Computational Performance

Scope:

- Calculate the expected computational efficiency of the job.
- Test the behavior of a program when executed in parallel.
- Scalability from 1...n processors.

Tools:

- **Speed Up**: factor that indicates the gain through parallelization.

\[ S(n) = \frac{T(1)}{T(n)} \]

- **Efficiency**: resource use efficient metric.

\[ E(n) = \frac{S(n)}{n} = \frac{T(1)}{n \cdot T(n)} \]

- Efficiency is archived when the measurement is kept at factor 0.5.
Computational Performance

$$S(n) = \frac{T(1)}{T(n)}$$

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)}$$

<table>
<thead>
<tr>
<th>N =</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>100/(40+20) = 1.666</td>
<td>2.5</td>
<td>3.333</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>100/(2*60) = 0.83 = 1.666/2</td>
<td>0.625</td>
<td>0.416</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Computational Performance

<table>
<thead>
<tr>
<th>Task</th>
<th>Task per Node</th>
<th>Time</th>
<th>Speedup</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1:00:27</td>
<td>1,0</td>
<td>1,0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0:33:47</td>
<td>1,8</td>
<td>0,9</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0:18:02</td>
<td>3,4</td>
<td>0,8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0:09:13</td>
<td>6,6</td>
<td>0,8</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>0:05:06</td>
<td>11,9</td>
<td>0,7</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>0:02:31</td>
<td>24,0</td>
<td>0,8</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>0:01:18</td>
<td>46,5</td>
<td>0,7</td>
</tr>
<tr>
<td>128</td>
<td>16</td>
<td>0:00:48</td>
<td>75,6</td>
<td>0,6</td>
</tr>
<tr>
<td>256</td>
<td>16</td>
<td>0:00:34</td>
<td>106,7</td>
<td>0,4</td>
</tr>
<tr>
<td>512</td>
<td>16</td>
<td>0:00:31</td>
<td>117,0</td>
<td>0,2</td>
</tr>
<tr>
<td>640</td>
<td>16</td>
<td>0:00:23</td>
<td>157,7</td>
<td>0,2</td>
</tr>
<tr>
<td>1024</td>
<td>32</td>
<td>0:00:22</td>
<td>164,9</td>
<td>0,2</td>
</tr>
</tbody>
</table>
Compilation

<table>
<thead>
<tr>
<th>option</th>
<th>optimization level</th>
<th>execution time</th>
<th>code size</th>
<th>memory usage</th>
<th>compile time</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O0</td>
<td>optimization for compilation time (default)</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-O1 or -O</td>
<td>optimization for code size and execution time</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-O2</td>
<td>optimization more for code size and execution time</td>
<td>---</td>
<td></td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>-O3</td>
<td>optimization more for code size and execution time</td>
<td>---</td>
<td></td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>-Os</td>
<td>optimization for code size</td>
<td>--</td>
<td></td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>-Ofast</td>
<td>O3 with fast none accurate math calculations</td>
<td>---</td>
<td></td>
<td>+</td>
<td>+++</td>
</tr>
</tbody>
</table>

`gcc -Olevel [options] [source files] [object files] [-o output file]`
Compilation Automation Tools

main.c
```c
#include "print.h"

int main() {
    printf(42);
}
```

print.h
```c
void printInt(int x);
```

print.c
```c
#include<stdio.h>

void printInt(int x) {
    printf("%d\n", x);
}
```

```make
- demo ls
Makefile main.c print.c print.h
- demo
- demo gcc -o main print.c main.c -I
- demo ls
Makefile main main.c print.c print.h
- demo ./main
  42
- demo
```

```make
- demo ls
Makefile main.c print.c print.h
- demo make
  gcc -c main.c
gcc -c print.c
gcc main.o print.o -o main
- demo ls
Makefile main main.o print.c print.h print.o
- demo ./main
  42
```
• **Make**
  - Carefully review of phase options
  
  `./configure --help`
  - Verbose mode `VERBOSE=1`
  - Use compile parallelism. `-j`

• **Guide when an error comes out**
  - Compile with single thread `--j1`
  - Always look for the first error in the list
  - Compile again
  - Google
  - Compile without optimizations
  - Patience.
You can use `/dev/shm` to improve the performance of application software in parallel tasks or overall Linux system performance. On heavily loaded system, it can make tons of difference.

- dd if=/dev/zero of=/dev/shm/test1.img bs=1G count=1
- dd if=/dev/zero of=$HOME/test1.img bs=1G count=1
- dd if=/dev/zero of=/tmp/test1.img bs=1G count=1
Disk Quota

Last login: Sun Nov 20 21:13:05 2022 from toctoc.sc3.uis.edu.co

SC3 UIS - GUANE

Centro de Supercomputacion y Calculo Cientifico
Universidad Industrial de Santander

Hostname: guane.uis.edu.co
Distribucion: CentOS Linux release 7.9.2009 (Core)
Kernel: 3.10.0-1160.76.1.el7.x86_64

System Load: 0.00, 0.01, 0.05
Memory Usage: 0.0%
Local Users: 2
Swap Usage: 1.3%
Whoami: cbernal

Disk Quota Usage:
 Filesystem  space  quota  limit
     /home   2636M  9216M  10240M

Para mas informacion de como utilizar la plataforma: http://wiki.sc3.uis.edu.co/
Para cambiar la clave de acceso asignada, digite el comando: passwd
In a HPC SysAmin team, Consultants handle thousands of support requests per year. In order to ensure efficient timely resolution of issues include as much of the following as possible when making a request.

* Error messages
* Job Ids
* Location of relevant files
* Input/output
* Job scripts
* Source code
* Executables
* Output of module list
* Any steps you have tried
* Steps to reproduce
Thank you