



Cluster de GPUs de la UAB Apuntes de usuario

*24 de Octubre de 2022
versión 0.1*

Guillermo Torres, Debora Gil

Contents

1	Acceso al Cluster	3
2	Instalación de un Entorno Virtual de Python	4
2.1	Crear un entorno virtual	4
2.2	Activar y desactivar el entorno virtual	4
2.3	Salvar un entorno virtual	5
2.4	Replicar un entorno virtual	5
2.5	Instalar PyTorch	5
2.6	Gestión de paquetes	5
2.7	Eliminar un entorno virtual	6
3	Ejecutar un procesos en el Cluster	6
3.1	Copiar ficheros De nuestro PC hacia el Cluster	6
3.2	Copiar ficheros del Cluster hacia nuestra PC	6
3.3	Permisos de ejecución	6
3.4	Ejecución de un proceso que usa GPU	6
3.4.1	Lanzar un proceso a la lista de jobs	7
3.4.2	Mostrar la lista de jobs	7
3.4.3	Cancelar un proceso	9
3.4.4	Obtener los resultados de nuestro proceso	9
3.5	Ejecutar un proceso que usa solo CPUs	9
4	Especificaciones del Cluster	10
5	Comandos básicos	10
5.1	Comprimir y descomprimir un directorio	10
6	Edición de archivos de texto	12
6.1	El editor Nano	12
6.2	El editor Vi	12

1 Acceso al Cluster

El login al cluster es a través de la IP 158.109.75.50 y el puerto 22.

Para sistemas operativos Windows se recomienda usar MobaXterm (<https://mobaxterm.mobatek.net>). Para iniciar una sesión, hay que ir al recurso de menú "Sessions" y seleccionar la opción "SSH". Se os abrirá una ventana de configuración de la sesión. Completad los campos del host (IP), puerto y vuestro usuario tal y como muestra la figura 1. La primera vez que iniciéis la sesión os pedirá el password.

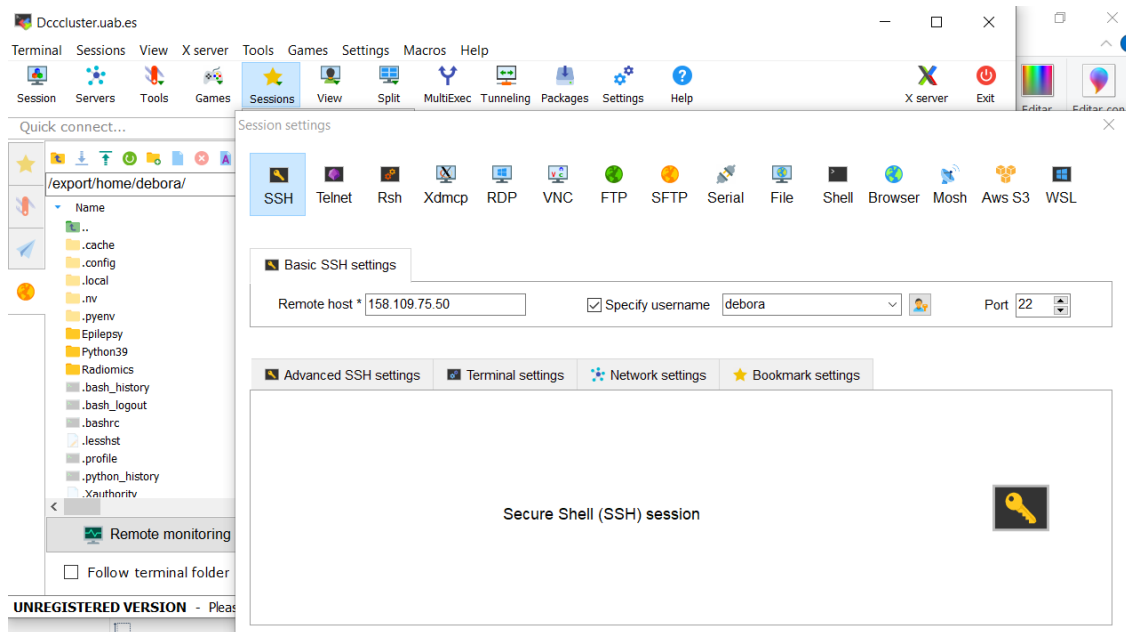


Figure 1: Configuración de una sesión SSH para conectarse al servidor con MobaXterm.

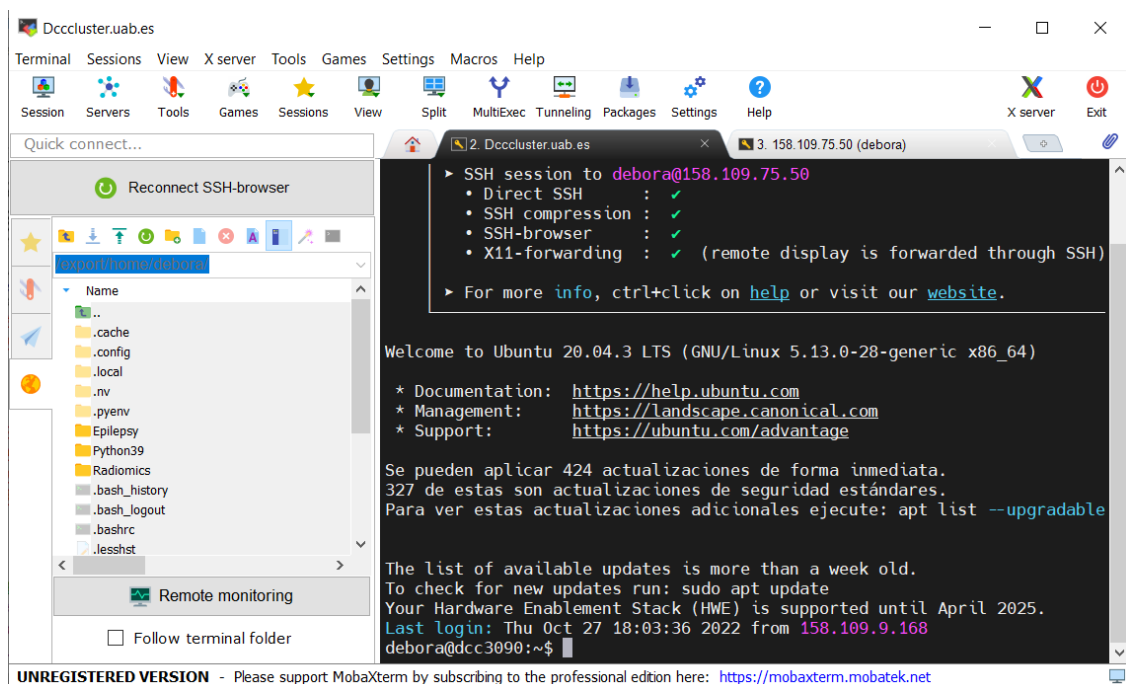


Figure 2: Sesión SSH con MobaXterm.

Una vez conectados, en la parte derecha (véase figura 2) se os muestra el árbol de directorios similar al del explorador de archivos. Además disponéis en la parte superior del browser

de botones para gestionar la creación de carpetas, así como, subir/bajar ficheros. Clicando con el botón derecho encima de un fichero, lo podeis editar usando el editor de MobaXTerm.

Para sistemas operativos basados en Unix, GNU/Linux o mac OS X se utiliza el comando ssh desde una terminal:

```
$ ssh -p 22 usuario@158.109.75.50
```

2 Instalación de un Entorno Virtual de Python

Para poder lanzar procesos en el cluster, cada usuario tiene que crear primero un entorno virtual de Python en el cluster, donde instalará los diferentes paquetes de versiones específicas. Para ello, hay que seguir estos pasos:

1. Crear el entorno virtual (sección 2.1).
2. Activar el entorno virtual (sección 2.2).
3. Replicar el entorno virtual que está en 'GPUServer_Requirements.tx' (sección 2.4).
4. Instalar Pytorch (sección 2.5).

2.1 Crear un entorno virtual

Para crear un entorno virtual de nombre, supongamos 'freeyourmind395', deberemos dirigirnos al directorio donde queremos que se cree dicho entorno y ejecutar:

```
$ python3 -m venv freeyourmind395
```

Esta instrucción nos creará la carpeta freeyourmind395 en el directorio donde la hayamos ejecutado. En esta carpeta es donde se instalarán todos los paquetes de python.

2.2 Activar y desactivar el entorno virtual

Una vez creado el entorno virtual, tendremos que activarlo para comenzar a instalar los paquetes y poder ejecutar los procesos. Para activarlo hay que ejecutar:

```
$ source freeyourmind395/bin/activate
```

A continuación el prompt nos muestra:

```
(freeyourmind395) user@dcc3090: $
```

que indica mediante la leyenda "(freeyourmind395)" que el entorno virtual está activo.

Y para desactivarlo solo bastará:

```
(freeyourmind395) usuario@dcc3090: $ deactivate
```

Y el prompt entonces muestra:

```
user@dcc3090: $
```

quitando la leyenda "(freeyourmind395)" que confirma que el entorno virtual ha sido desactivado.

2.3 Salvar un entorno virtual

Este paso ya lo hemos (la cátedra) realizado para generar el fichero 'GPUServer_Requirements.txt' por lo que no tiene que hacerlo. En nuestra PC, salvamos los paquetes instalados de nuestro entorno virtual en el fichero 'GPUServer_Requirements.txt':

```
$ pip freeze > GPUServer_Requirements.txt
```

2.4 Replicar un entorno virtual

Copie en el cluster el fichero 'GPUServer_Requirements.txt' y ejecute el siguiente comando para instalar el listado de paquetes que contiene dicho fichero:

```
$ pip install -r GPUServer_Requirements.txt
```

2.5 Instalar PyTorch

El Cluster tiene instalado un driver de CUDA en su Versión 11.6, que lo muestra el comando **nvidia-smi** (véase figura 7) en la parte superior de la salida. Y para instalar la versión de Pytorch acorde a dicho driver de CUDA, según <https://pytorch.org/get-started/locally/>, hay que ejecutar:

```
$ pip3 install torch torchvision torchaudio --extra-index-url  
https://download.pytorch.org/whl/cu116
```

2.6 Gestión de paquetes

Recordar activar previamente el entorno virtual antes de comenzar.
Para instalar un paquete:

```
$ pip install numpy
```

Para averiguar las versiones disponibles de un paquete:

```
$ pip install numpy==
```

Para instalar una versión específica de un paquete:

```
$ pip install numpy==1.19.2
```

Para actualizar un paquete que ya tenemos instalado:

```
$ pip install --upgrade numpy
```

Para conocer todos los paquetes instalados en el entorno virtual:

```
$ pip list
```

Para eliminar un paquete:

```
$ pip uninstall numpy
```

2.7 Eliminar un entorno virtual

Cada entorno virtual es creado en un directorio y en su interior se instalan todos los paquetes. Así que para eliminar un entorno virtual basta con borrar dicho directorio. Por ejemplo, si el entorno virtual fue nombrado 'myvirtenv' y está ubicado en /home/user, entonces para eliminarlo definitivamente del sistema de ficheros, se ejecuta:

1. `$ cd /home/user`
2. `$ rm -rf freeyourmind395`

3 Ejecutar un procesos en el Cluster

Todos los procesos que usan una o más GPUs no se ejecutan directamente, sino que deben agregarse a una lista de jobs y esperar allí hasta que le sea asignado una GPU (o las GPUs que haya solicitado) para su ejecución. Para hacerlo, siga estos pasos:

1. Copiar en el Cluser los ficheros: process2jobList.sh, testPyTorch.py (sección 3.1).
2. Asignarle permisos de ejecución a 'process2jobList.sh' (sección 3.3).
3. Lanzar el proceso 'testPyTorch.py' agregándolo a la lista de jobs a través de 'process2jobList.sh' (sección 3.4.1).
4. Ver la salida producida por la ejecución de 'testPyTorch.py' (sección 3.4.4).

3.1 Copiar ficheros De nuestro PC hacia el Cluster

Para copiar recursivamente el directorio localFolderPC de nuestro PC hacia el cluster, ejecutar desde una terminal en nuestro PC:

```
$ scp -P 22 -rp localFolderPC user@158.109.75.50:/home/user/remoteFolderCluster
```

3.2 Copiar ficheros del Cluster hacia nuestra PC

Para copiar el directorio 'remoteFolderCluster' que está en el cluster hacia nuestro PC ejecutar desde una terminal (en nuestro PC):

```
$ scp -P 22 -rp user@158.109.75.50:/home/user/remoteFolderCluster .
```

3.3 Permisos de ejecución

Para poder ejecutar el fichero 'process2jobList.sh', antes debemos asignarle permisos de ejecución de la siguiente forma:

```
$ chmod +x process2jobList.sh
```

3.4 Ejecución de un proceso que usa GPU

Los procesos que usan una o más GPUs deben enviarse a una **lista de jobs**, que aplica una política FIFO (First In First Out) por prioridades para asignar las GPUs a los procesos que están en dicha lista.

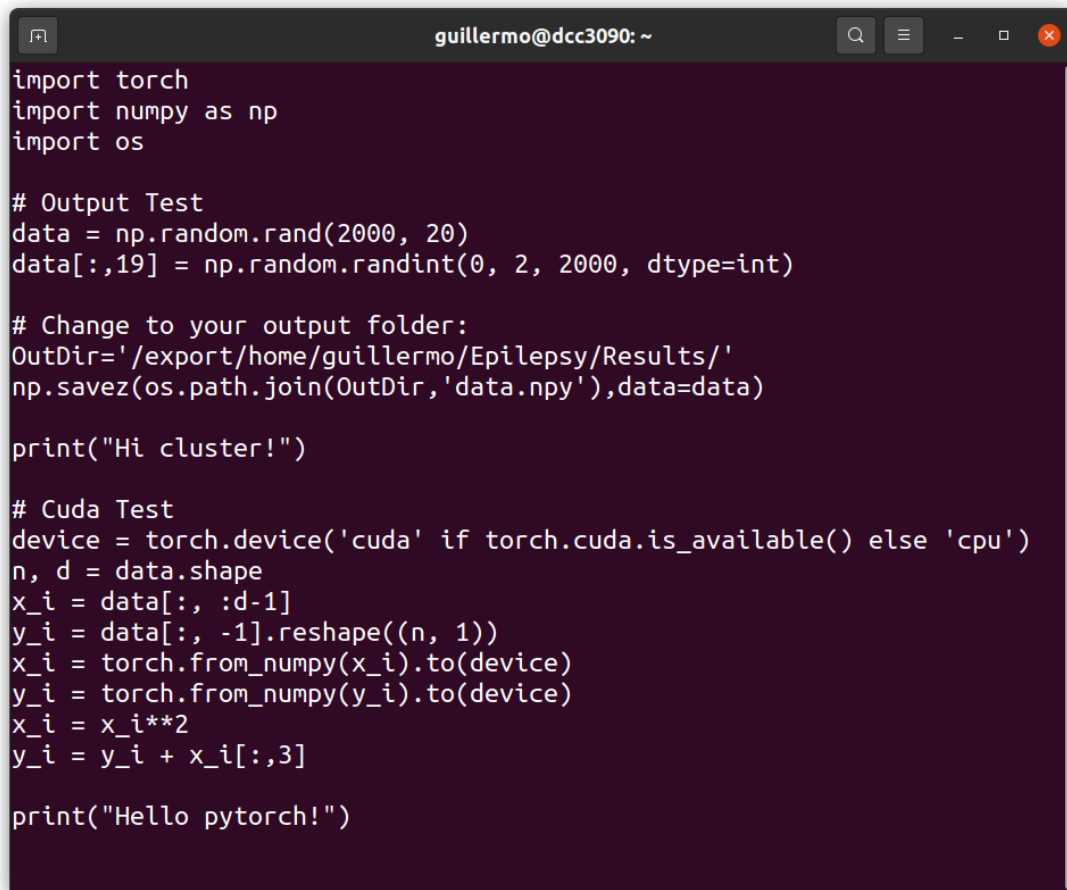
A terminal window with a dark background and light-colored text. The window title is 'guillermo@dcc3090: ~'. It displays the content of a Python script named 'testPyTorch.py'. The script imports 'torch', 'numpy as np', and 'os'. It defines an 'Output Test' section where it generates random data and saves it to a file. It then prints 'Hi cluster!'. A 'Cuda Test' section follows, which sets the device to 'cuda' if available, otherwise 'cpu'. It reshapes the data, converts it to PyTorch tensors, performs a calculation (x_i squared plus y_i), and prints 'Hello pytorch!'.

Figure 3: El comando cat muestra por pantalla el contenido del fichero 'testPyTorch.py'.

3.4.1 Lanzar un proceso a la lista de jobs

Para ejecutar nuestro proceso 'testPyTorch.py' usaremos un script bash que es el 'process2jobList.sh' que tiene indicaciones que son interpretadas por el **gestor de colas** que tiene el Cluster. Los ficheros 'testPyTorch.py' y 'process2jobList.sh' se muestran en las figuras 3 y 4 respectivamente. Note que en la última línea de 'process2jobList.sh' se indica el fichero que queremos ejecutar indicando python y la ruta absoluta del fichero /home/user/testPyTorch.py.

Para lanzar nuestro proceso y que sea agregado en la lista de jobs, primero debemos tener activado el entorno virtual (sección 2.2) y entonces ejecutamos los siguiente:

```
$ sbatch process2jobList.sh
```

3.4.2 Mostrar la lista de jobs

Para mostrar todos los procesos que están en la lista de jobs usamos:

```
$ squeue
```

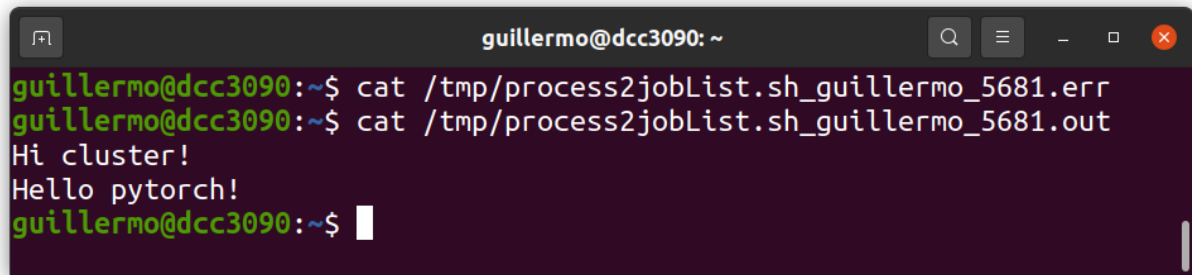
Una vez agregado nuestro proceso a la lista de jobs, el mismo aguardará allí en el estado W (Waiting), hasta que el workload manager le asigne los recursos de hardware necesarios para su ejecución, momento en el cual su estado cambia a R (Running), es decir en ejecución, y a partir de ese momento nuestro proceso comienza realmente a ejecutarse. La figura 5 muestra la secuencia de ejecución de los comandos sbatch y squeue.

```
guillermo@dcc3090: ~  
guillermo@dcc3090:~$ cat process2jobList.sh  
#!/bin/bash  
#SBATCH -n 4 # Number of CPU's cores. Maximum 10 CPU's cores.  
#SBATCH -N 1 # Ensure that all cores are on one machine  
#SBATCH -D /tmp # Working directory  
#SBATCH -t 2-00:05 # Runtime in D-HH:MM  
#SBATCH -p dcc # Partition to submit to  
#SBATCH --mem 2048 # request 2GB of RAM memory. Maximum 60 GB.  
#SBATCH -o %x_%u_%j.out # File to which STDOUT will be written  
#SBATCH -e %x_%u_%j.err # File to which STDERR will be written  
#SBATCH --gres gpu:1 # request 1 GPU. Maximum 8 GPUs  
sleep 1  
python /export/home/guillermo/Epilepsy/Code/testPyTorch.py  
guillermo@dcc3090:~$
```

Figure 4: El comando `cat` muestra por pantalla el contenido del fichero `'process2jobList.sh'` que en su última línea tiene el fichero que queremos ejecutar.

```
(.freeyourmind395) guillermo@dcc3090:~$ sbatch process2jobList.sh  
Submitted batch job 5681  
(.freeyourmind395) guillermo@dcc3090:~$ squeue  
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)  
      5681      dcc process2 guillerm R        0:00      1 dcc3090  
      5631      dcc job_new_   yyang  R  4-00:24:23      1 dcc3090  
(.freeyourmind395) guillermo@dcc3090:~$
```

Figure 5: **sbatch** agrega nuestro proceso `'process2jobList.sh'` a la **lista de jobs** con el job-id 5681. **squeue** nos muestra que hay dos jobs en la lista de jobs, en la columna ST (state) se indica R (Running) para los procesos en ejecución o W (Waiting) para los procesos que esperan recibir los recursos de hardware, y TIME indica el tiempo transcurrido del proceso en dicha lista.

A terminal window titled 'guillermo@dcc3090: ~' with search, menu, and window control icons. It shows the execution of 'cat /tmp/process2jobList.sh_guillermo_5681.err' and 'cat /tmp/process2jobList.sh_guillermo_5681.out'. The output of the second command is 'Hi cluster!' and 'Hello pytorch!'.

```
guillermo@dcc3090:~$ cat /tmp/process2jobList.sh_guillermo_5681.err
guillermo@dcc3090:~$ cat /tmp/process2jobList.sh_guillermo_5681.out
Hi cluster!
Hello pytorch!
guillermo@dcc3090:~$
```

Figure 6: El comando 'cat' muestra por pantalla la salida producida por el fichero 'process2jobList.sh'.

3.4.3 Cancelar un proceso

Para cancelar la ejecución de nuestro proceso usamos el id-job que le fue otorgado cuando lo lanzamos. Si desconocemos este id, bastará con ejecutar 'squeue' para averiguarlo. Entonces ejecutar:

```
$ scancel 5636
```

3.4.4 Obtener los resultados de nuestro proceso

El proceso 'testPyTorch.py' genera las siguientes salidas:

- salva un fichero numpy en /export/home/guillermo/Epilepsy/Results/data.npy, y
- la salida por pantalla es salvada en estos ficheros:
 - /tmp/process2jobList.sh_guillermo_5681.err
 - /tmp/process2jobList.sh_guillermo_5681.out

Estos últimos dos ficheros de arriba se salvan en el directorio "/tmp" porque así está especificado en 'process2jobList.sh' en la línea:

```
#SBATCH -D /tmp # Working directory
```

De manera que si deseais salvar estos ficheros en otra ubicación, bastará con modificar dicha ruta. Los errores de ejecución se salvan en 'process2jobList.sh_guillermo_5681.err'. Y la salidas por pantalla se salvan en 'process2jobList.sh_guillermo_5681.out'.

Con el comando cat podemos mostrar los contenidos de estos ficheros, como se observa en la figura 6. El fichero process2jobList.sh_guillermo_5681.err está vacío porque no se ha producido ningún error y process2jobList.sh_guillermo_5681.out tiene "Hi cluster" y "Hello pytorch!" que han sido producidas por las instrucciones de print en testPyTorch.py.

3.5 Ejecutar un proceso que usa solo CPUs

Se ejecutan directamente sin necesidad de enviarse a una lista de jobs y se hace del siguiente modo (recuerda activar el entorno virtual previamente):

```
$ python computosxCPU.py
```

O en caso de ser un fichero ejecutable (debe tener permisos de ejecución):

```
$ ./miEjecutable.py
```

Es importante señalar que si nuestro proceso está ejecutándose y finalizamos nuestra sesión en el Cluster antes que dicho proceso finalice su ejecución, entonces ese proceso es matado inmediatamente. Por ello, para que nuestro procesamiento continúe su ejecución aún cuando nos hemos deslogueado del Cluster, una solución sencilla es lanzarlo en background (segundo plano) del siguiente modo:

```
$ ./miEjecutable.py &
```

Aunque este tópico está más allá del alcance de esta guía, no queremos dejar de mencionar una solución alternativa que es comúnmente usada y que es usar el software `tmux` (un multiplexador de terminales que está instalado en el Cluster) que permite ejecutar procesos sin que se vean finalizados (matados) porque nos hemos deslogueado del Cluster.

4 Especificaciones del Cluster

El cluster está conformado por un nodo con:

- 96 CPUs de arquitectura x64. Cada CPU tiene un total de 24 cores, que además cuenta con el `hyperthreading` habilitado a 2 hilos por core. Solicitud máxima de 10 cores por proceso.
- 512 GB de memoria RAM. Solicitud máxima de 60 GB por proceso.
- 3,7 TB de disco duro, con una cuota de 150 GB por usuario.
- 1 GB de conexión de red, y
- 8 GPUs. El comando **`nvidia-smi`** muestra por pantalla un listado de las GPUs que tiene el Cluster, véase figura 7.

Es importante señalar que los recursos de hardware que solicita nuestro proceso para su ejecución, definidos en el fichero `process2jobList.sh`, se ajustan al uso de hardware que realmente requiere dicho proceso para su ejecución. Caso contrario, se nos asignarán más recursos de los que necesitamos y que serán retenidos por nuestro proceso hasta su finalización.

5 Comandos básicos

La tabla 1 contiene una lista de comandos básicos.

5.1 Comprimir y descomprimir un directorio

Para comprimir el directorio "Code" que incluya el contenido de sus subdirectorios:

```
$ zip -r code.zip Code
```

Y para descomprimirlo:

```
$ unzip code.zip
```

```

guillermo@dcc3090: ~/example
Thu Nov  3 13:17:29 2022

+-----+
| NVIDIA-SMI 510.47.03      Driver Version: 510.47.03      CUDA Version: 11.6      |
+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                               |                      |              MIG M. |
+-----+
| 0  NVIDIA GeForce ...  On      | 00000000:01:00.0 Off |           N/A       |
| 80%   74C    P2     332W / 350W | 23910MiB / 24576MiB |    96%    Default   |
|                               |                      |              N/A     |
+-----+
| 1  NVIDIA GeForce ...  On      | 00000000:25:00.0 Off |           N/A       |
| 82%   75C    P2     329W / 350W | 23922MiB / 24576MiB |    94%    Default   |
|                               |                      |              N/A     |
+-----+
| 2  NVIDIA GeForce ...  On      | 00000000:41:00.0 Off |           N/A       |
| 77%   72C    P2     325W / 350W | 23128MiB / 24576MiB |    95%    Default   |
|                               |                      |              N/A     |
+-----+
| 3  NVIDIA GeForce ...  On      | 00000000:61:00.0 Off |           N/A       |
| 64%   63C    P2     318W / 350W | 23910MiB / 24576MiB |    92%    Default   |
|                               |                      |              N/A     |
+-----+
| 4  NVIDIA GeForce ...  On      | 00000000:81:00.0 Off |           N/A       |
| 30%   30C    P8       27W / 350W |      1MiB / 24576MiB |     0%    Default   |
|                               |                      |              N/A     |
+-----+
| 5  NVIDIA GeForce ...  On      | 00000000:A1:00.0 Off |           N/A       |
| 30%   30C    P8       27W / 350W |      1MiB / 24576MiB |     0%    Default   |
|                               |                      |              N/A     |
+-----+
| 6  NVIDIA GeForce ...  On      | 00000000:C1:00.0 Off |           N/A       |
| 30%   30C    P8       20W / 350W |      1MiB / 24576MiB |     0%    Default   |
|                               |                      |              N/A     |
+-----+
| 7  NVIDIA GeForce ...  On      | 00000000:E1:00.0 Off |           N/A       |
| 30%   27C    P8       30W / 350W |      1MiB / 24576MiB |     0%    Default   |
|                               |                      |              N/A     |
+-----+

+-----+
| Processes: |
| GPU  GI    CI          PID    Type   Process name                      GPU Memory |
|   ID   ID    ID              |    |                  |      Usage |
+-----+
|  0    N/A  N/A       440176     C   ...3/envs/pytorch/bin/python     23907MiB |
|  1    N/A  N/A       440177     C   ...3/envs/pytorch/bin/python     23919MiB |
|  2    N/A  N/A       440178     C   ...3/envs/pytorch/bin/python     23125MiB |
|  3    N/A  N/A       440179     C   ...3/envs/pytorch/bin/python     23907MiB |
+-----+

guillermo@dcc3090:~/example$

```

Figure 7: Listado de GPUs: la primer columna 'GPU Fan' indica el ID de la GPU que está en el rango de 0-7 y debajo está el uso (en porcentaje) del ventilador de la GPU. La columna 'Volatile GPU-Util' indica en porcentaje el uso de la GPU. Por ejemplo, la GPU con ID 0 tiene su ventilador al 80% y está siendo utilizada al 96% de su capacidad de cómputo, mientras que las GPUs 4 a la 7 no están siendo utilizadas y muestran un 0% de uso.

6 Edición de archivos de texto

Para entornos Windows, MobaXterm permite editar ficheros directamente sin necesidad de usar los programas que se indican en esta sección. En caso de entornos UNIX, GNU/Linux y OS X, para editar un fichero en el Cluster es necesario hacerlo a través de un editor de texto como Nano o Vi.

Una alternativa, que es solo para entornos UNIX, GNU/Linux y OS X, es copiar el fichero del Cluster a la PC y una vez modificado localmente subirlo al Cluster, para esto puede ver la sección 3.2 y 3.1. Esta opción no puede realizarse en entornos Windows, ya que de hacerlo, en su PC cuando modifique el fichero, también se agregan caracteres no visibles al fichero y cuando se intentan ejecutar en el Cluster (con GNU/Linux) se producirá un error porque no lo puede interpretar. Estos caracteres invisibles agregados en entornos Windows se pueden visualizar en el editor Vi y Nano.

6.1 El editor Nano

Para crear o modificar el fichero de texto 'clasificar.py' usando el editor de texto Nano, en una terminal ejecutar:

```
$ nano clasificar.py
```

Luego de modificar el texto, para salir debe presionar las teclas Ctrl + x. El símbolo ^ en Nano representa a la tecla Ctrl.

6.2 El editor Vi

El editor de texto "Vi" tiene dos estados que son: modo edición y modo comandos. Siempre está en uno u otro estado. Por defecto este editor ingresa en modo de comandos. La siguiente es una secuencia para editar un fichero 'clasificar.py':

1. \$ vi clasificar.py
2. presionar la tecla **i** (insert text) para ingresar en modo texto,
3. modificar el texto,
4. presionar la tecla **esc** para ingresar nuevamente en modo comandos, y
5. tipear **:x** para guardar y salir.

Table 1: Comandos básicos.

Comandos	Descripción
lsb_release -a	Información del sistema operativo, versión y release.
mem -g	Memoria RAM en Gigabytes.
ls -l	Lista los ficheros y directorios
pwd	Ruta del directorio de trabajo actual.
mkdir newDirectory	Crear un nuevo directorio.
rm -f file	Elimina el fichero file.
rm -rf oldDirectory	Elimina el directorio y todo su contenido recursivamente.
cp file1 file2	Copia el fichero file1 con el nombre file2 .
mv dir1/file dir2/file	Mueve el fichero file del directorio dir1 al dir2.
du -sh Directorio	Cantidad total de espacio que ocupa el 'Directorio' en el disco duro .
history	Historial de comandos ejecutados.