Cluster de cómputo III-LIDI

III-LIDI

Instituto de Investigación en Informática - LIDI

COMISIÓN DE INVESTIGACIONES CIENTÍFICAS

Facultad de INFORMÁTICA
UNIVERSIDAD NACIONAL DE LA PLATA

Tabla de contenidos

1 Introducción al cluster de cómputo III-LIDI

El Instituto de Investigación en Informática LIDI posee un **Cluster de Cómputo** que se accede remotamente y es utilizado por Alumnos, Docentes e Investigadores en el área de cómputo de altas prestaciones (HPC por sus siglas en inglés).

1.1 Características del cluster

El **Cluster de Cómputo** está organizado en 3 particiones:

Cluster Multicore (partición Blade): es un cluster conformado por 16 nodos. Cada nodo posee 8GB de RAM y 2 procesadores Intel Xeon E5405 de cuatro 4 cores cada uno que operan a 2.0GHz.

Intel Manycore (partición XeonPHI): es un equipo Manycore Intel Xeon PHI KNL (Knights Landing) 7230 que posee 128GB de RAM y 64 cores que operan a 1.3GHz.

Cluster MultiGPU (partición GPUS): es un cluster conformado por 2 nodos. Cada nodo posee 16GB de RAM y 2 GPUs Nvidia GeForce GTX 960 (Arquitectura Maxwell GM206). Cada GPU tiene capability 5.2, posee 2GB de RAM GDDR5 y 1024 cuda cores distribuidos en 8 SMMs.

La puerta de entrada al Cluster de Cómputo es el siguiente equipo:

Frontend: es un equipo que centraliza los servicios y permite enviar a ejecutar aplicaciones sobre las particiones descriptas anteriormente.

Cada equipo del Cluster de Cómputo se encuentra interconectado por una red Ethernet a 1Gbit.

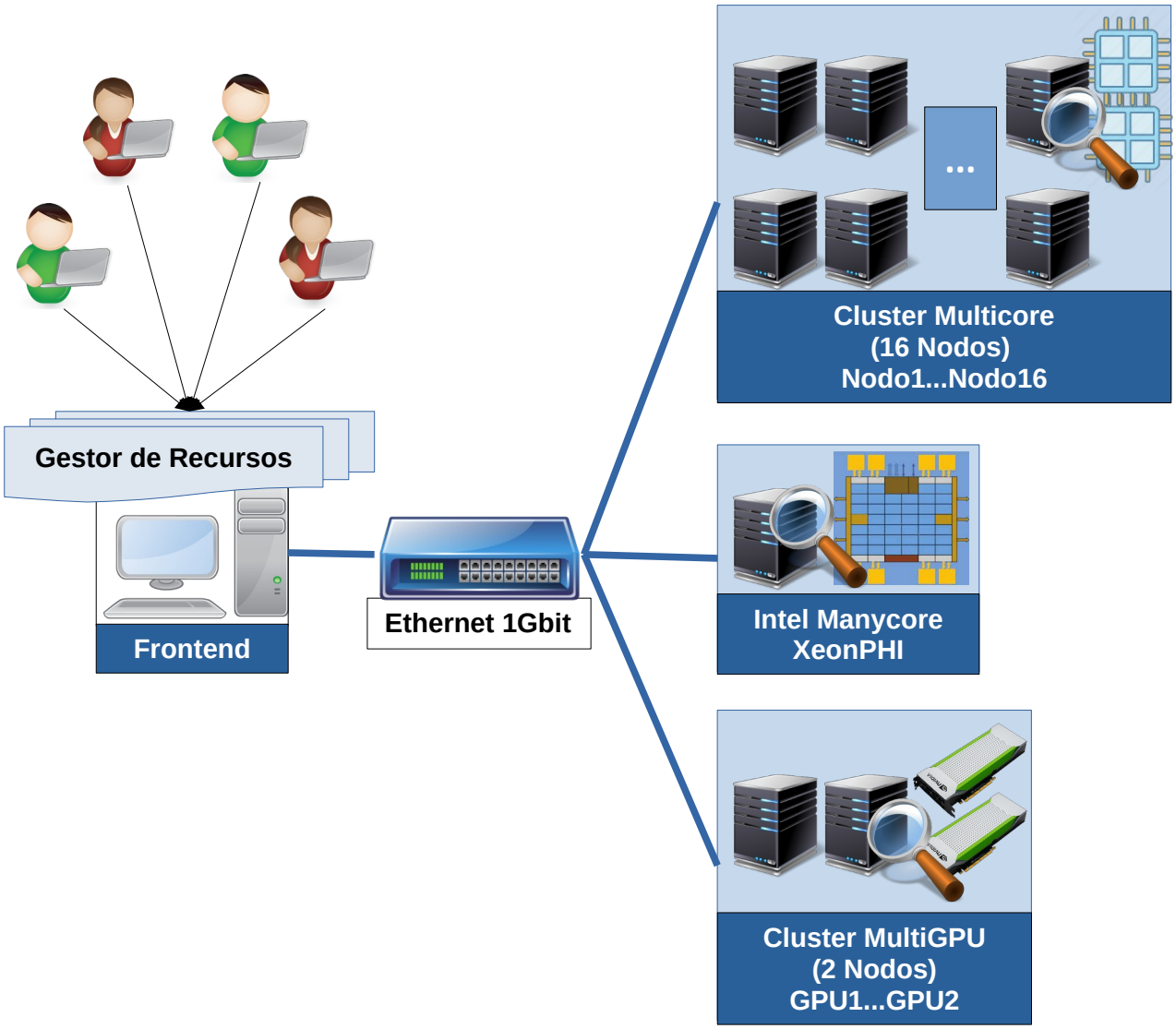
1.2 Modo de ejecución sobre el cluster

Para poder ejecutar sus aplicaciones, cada usuario debe acceder remotamente a través del Frontend.

Es importante entender, que el cluster es un recurso compartido por varios usuarios. Cada usuario debe ejecutar sus aplicaciones sobre el mismo de forma exclusiva, es decir sin que la ejecución de otro usuario interfiera. Para evitar conflictos entre las distintas ejecuciones, el cluster posee un gestor de recursos y cada usuario deberá ejecutar sus aplicaciones a través de éste.

1.3 Diagrama general del cluster

A continuación se muestra el diagrama general del cluster de cómputo.



1.4 Pasos para ejecutar aplicaciones sobre el cluster

Los pasos que deberá seguir cada usuario para ejecutar una aplicación sobre alguno de los equipos del cluster de cómputo son:

1. **Acceso remoto:** el cluster de cómputo se accede de forma remota a través del protocolo SSH. Este protocolo de red permite el acceso seguro a un equipo remoto. La forma de acceder dependerá de cada Sistema Operativo. En este paso podemos distinguir dos actividades:

   I. **Transferencia de archivos:** consiste en acceder de forma remota al Frontend para transferir archivos desde la máquina del usuario al cluster de cómputo. Generalmente, nos referimos a la transferencia del código fuente de la aplicación.

   II. **Acceso a consola de texto:** una vez transferidos los archivos, necesitamos abrir una consola de texto en el Frontend para ejecutar los comandos que nos permitirán la ejecución de aplicaciones sobre el cluster de cómputo.

2. **Ejecución:** consiste en la eventual compilación del código y la ejecución de la aplicación a través del gestor de recursos. La forma de ejecutar dependerá de los equipos elegidos (Cluster Multicore, Intel Manycore o Cluster MultiGPU) y del tipo de ejecución (Secuencial o Paralela)

3. **Resultados:** consiste en recuperar los resultados de la ejecución de la aplicación de acuerdo a como los retorna el gestor de recursos.

2 Acceso remoto

Como mencionamos anteriormente, para ejecutar aplicaciones sobre el cluster de cómputo necesitamos acceder al Frontend para transferir archivos y abrir una consola de texto para ejecutar comandos.

Realizamos la transferencia de archivos mediante el protocolo ***SCP***, un protocolo de red basado en ***SSH*** diseñado para transferencias de archivos seguras.

Una vez que se copiaron los archivos necesarios al cluster de cómputo, debemos acceder a una consola de texto en el Frontend para eventualmente compilar y posteriormente enviar a ejecutar. Para esto, utilizaremos el protocolo ***SSH***.

NOTA IMPORTANTE: no es posible ejecutar en los equipos del cluster de cómputo binarios compilados en otras máquinas. Las aplicaciones deberán compilarse en el cluster de cómputo.

La forma de usar estos protocolos varía de un sistema operativo a otro, ya sea Windows o Linux. A su vez, varía entre las distintas distribuciones de Linux.

2.1 Acceso desde Windows

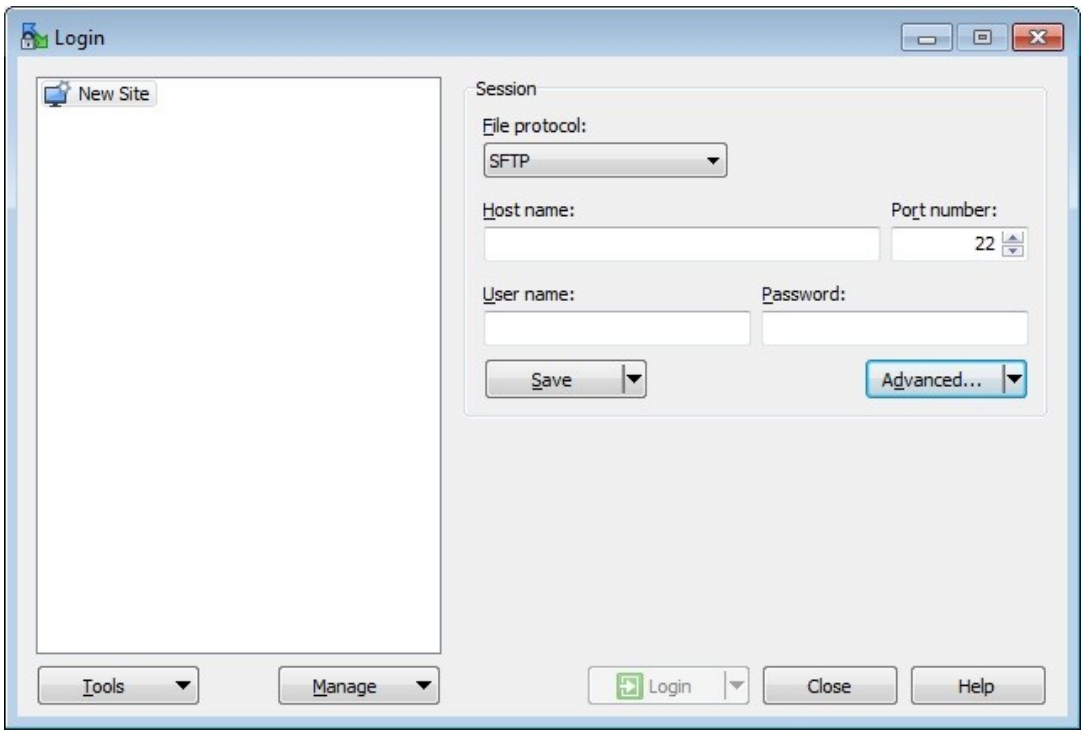
Para la transferencia de archivos sobre Windows, utilizaremos la herramienta ***WinSCP***. Descargar esta aplicación de https://winscp.net

Para acceder a la consola de texto, utilizaremos la herramienta ***Putty***. Descargar esta aplicación del sitio https://www.chiark.greenend.org.uk

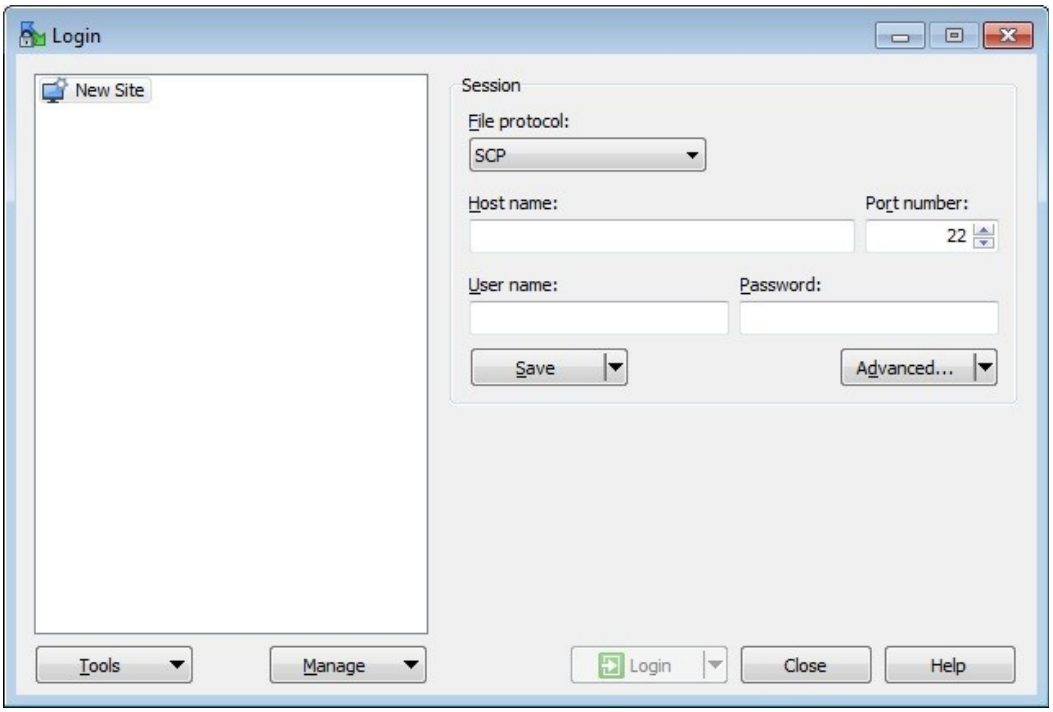
*NOTA IMPORTANTE: Descargar el archivo putty.exe (NO el instalador). Luego, copiar este archivo dentro de la carpeta donde instalamos **WinSCP**.*

2.1.1 Transferencia de archivos

Ejecutamos WinSCP y nos abre la pantalla inicial:

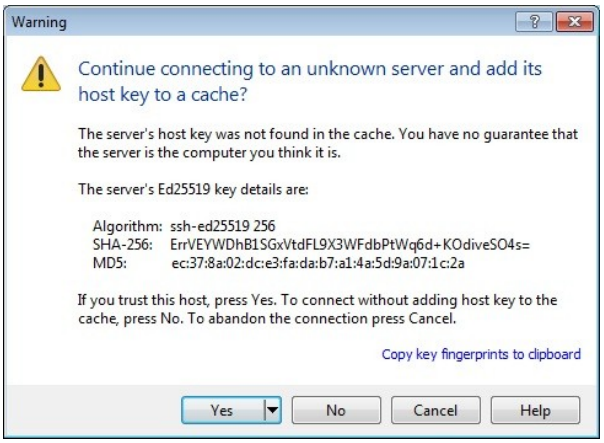


Debemos seleccionar SCP como protocolo de transferencias de archivos. Para esto desplegamos las opciones en File protocol y seleccionamos SCP. La pantalla inicial debería quedar entonces:



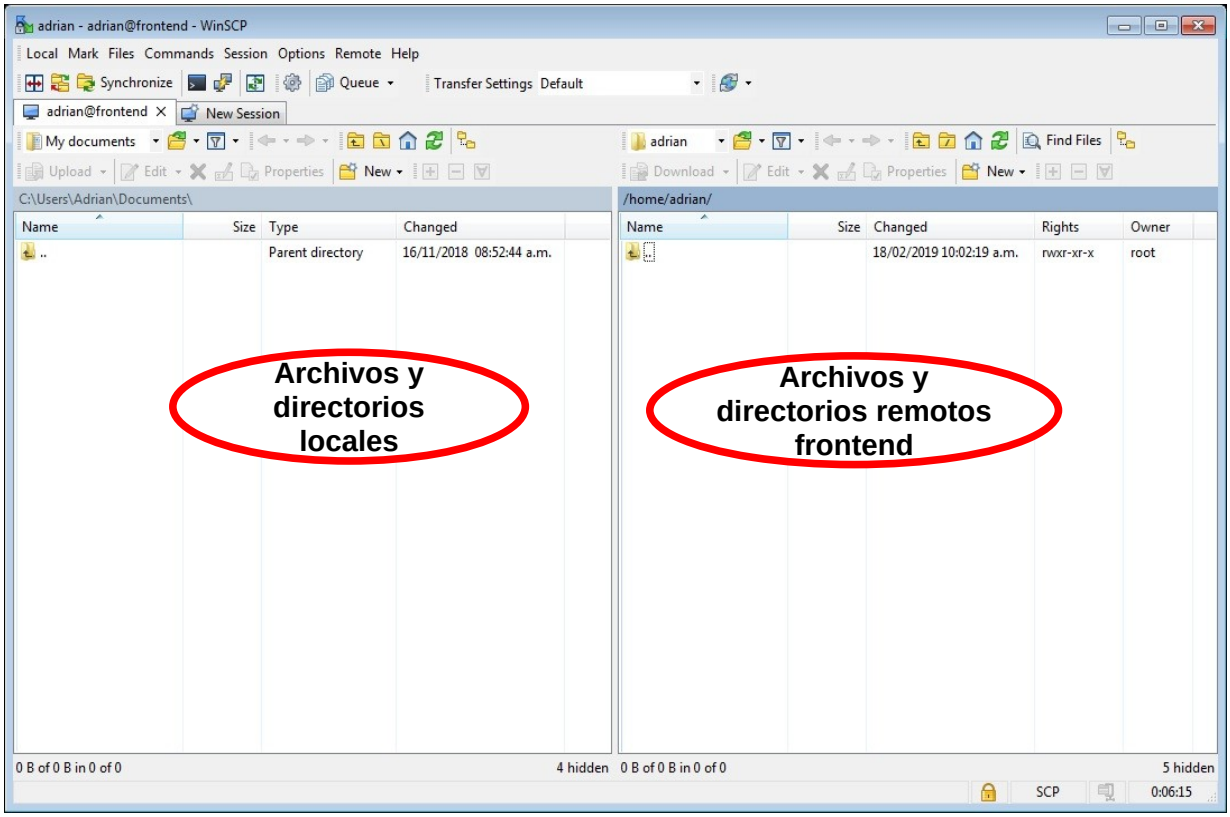
Luego, completar los campos de ***Host name***, ***User name*** y ***Password*** de acceso al frontend con los datos dados oportunamente por la cátedra. Finalizar presionando el botón **Login**. Podemos guardar las conexiones (botón **Save**) para no realizar una y otra vez el procedimiento de conexión.

Posiblemente, por primera vez, la autenticación retorne una o dos ventanas de Warning:



Por cada ventana presionar el botón **Yes**.

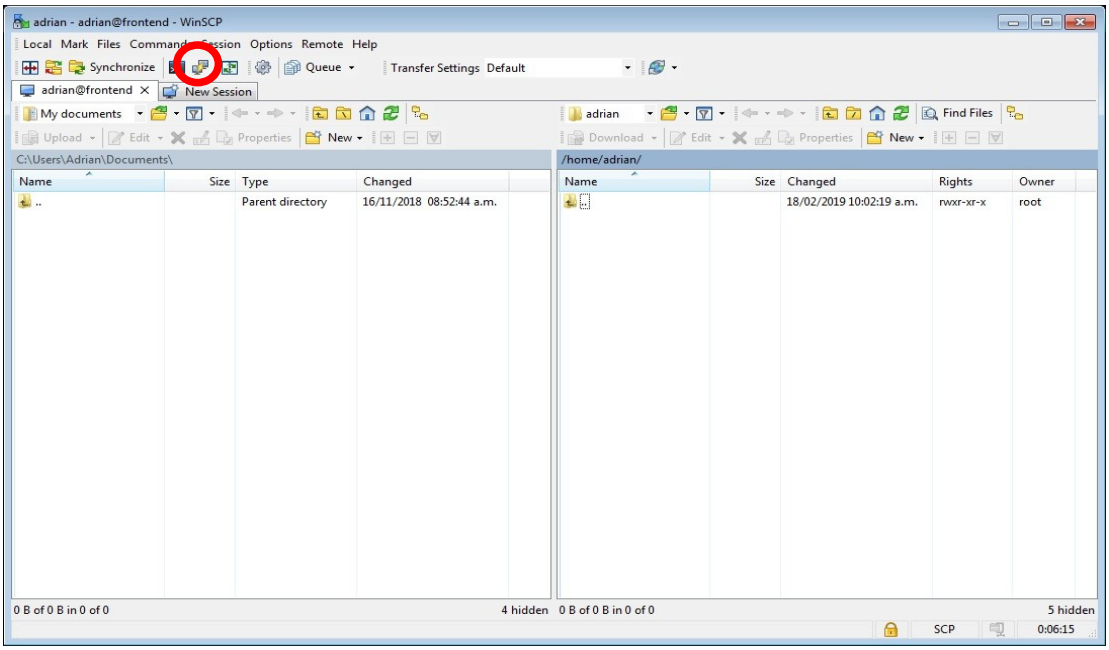
Si la autenticación fue exitosa se abrirá un administrador de archivos:



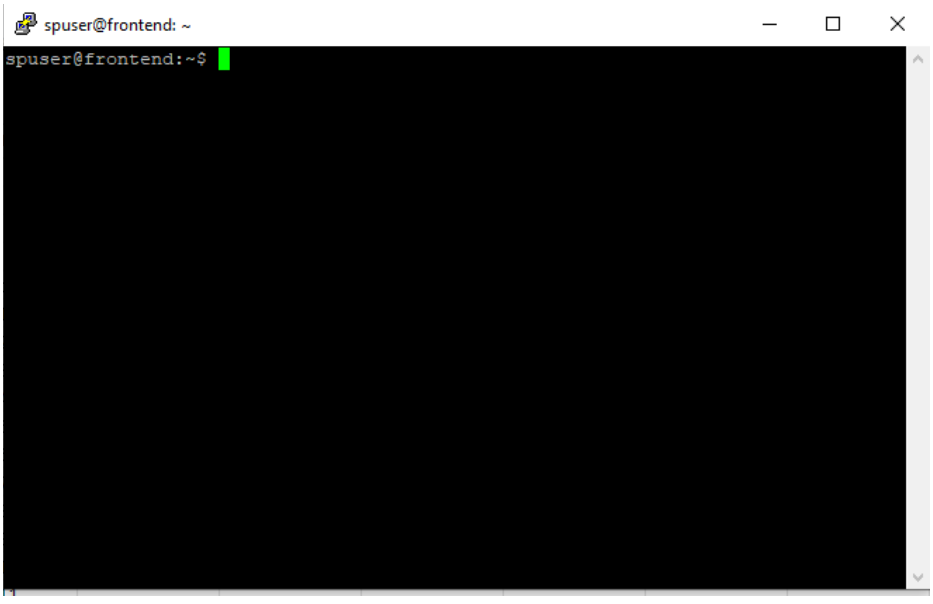
En el panela izquierdo se mostrarán los archivos locales y en el panel derecho se mostrarán los archivos en el Frontend. Se pueden transferir archivos y directorios entre la máquina local y el Frontend con sólo arrastrarlos.

2.1.2 Acceso a consola de texto

Podemos acceder a Putty a través de la aplicación WinSCP mediante el ícono **“Open Session in PuTTY”** tal como se indica en la siguiente imagen.



Se abrirá una consola de texto y se nos pedirá la contraseña del usuario del Frontend. Si la autenticación fue exitosa, nos encontraremos en una sesión SSH remota, es decir los comandos que ejecutemos en esa consola se ejecutarán en el Frontend.



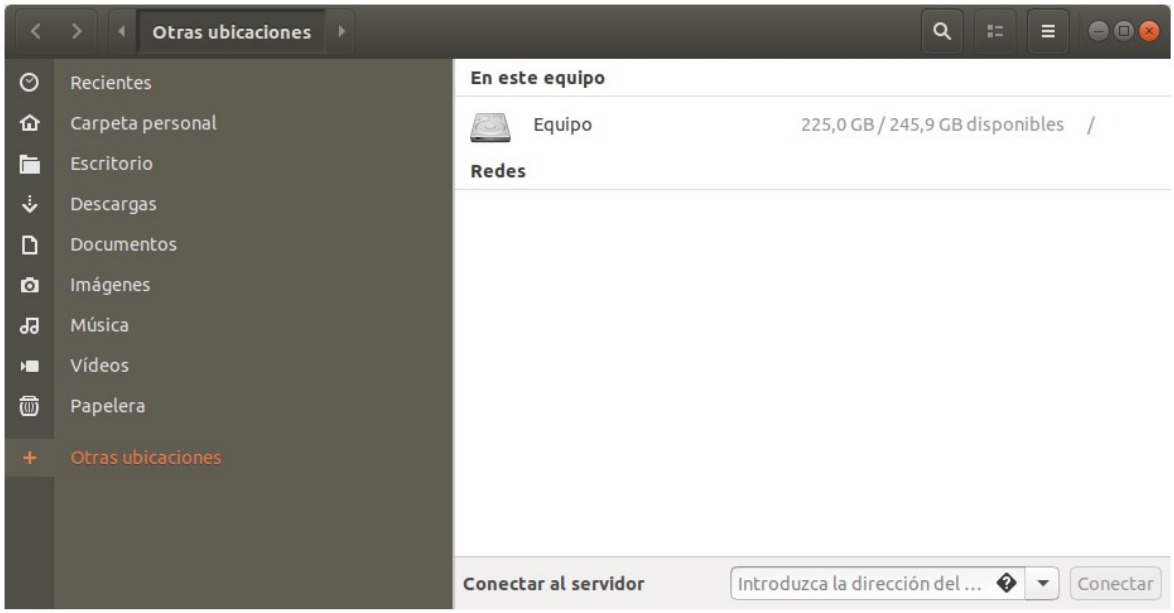
2.2 Acceso desde Linux

Las distintas distribuciones de Linux incorporan herramientas para utilizar los protocolos SCP y SSH. Por lo tanto, no es necesario instalar aplicaciones adicionales. Sin embargo, el uso de estos protocolos varía de una distribución de Linux a otra. Para nuestro ejemplo, utilizamos la distribución de Linux Ubuntu.

2.2.1 Transferencia de archivos

Para transferir archivos hacia el cluster de cómputo debemos abrir el navegador de archivos de Ubuntu (Nautilus).

En el lateral izquierdo, vamos a **"+ Otras ubicaciones"**. Deberíamos ver algo como:



Luego completar el campo de la parte inferior, donce dice **Conectar al servidor** con la siguiente información:

`ssh://HostNameFrontend/HOMEfrontendUSER`

HostNameFrontend es el nombre de host del Frontend.

HOMEfrontendUSER es el directorio del usuario con el que se ingresa al Frontend.

Se abrirá una ventana en la que pedirá usuario y contraseña, se deberá ingresar el **usuario** y **contrañeña** del Frontend.

Todos estos datos serán dados oportunamente por la cátedra.

Por último, presionar el botón conectar.

Una vez finalizado, el contenido del navegador de archivos será el contenido del directorio **HOMEfrontendUSER.**

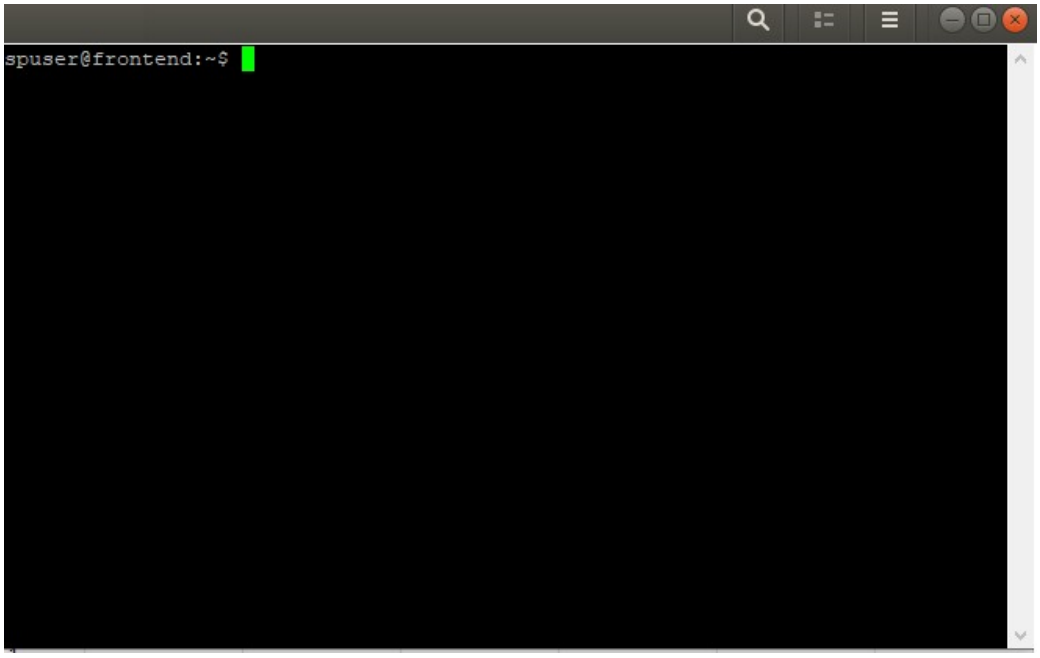
Se pueden transferir archivos y directorios entre la máquina local y el Frontend con sólo arrastrarlos desde una ventana del navegador de archivos local a la ventana del navegador de archivos en el Frontend.

2.2.2 Acceso a consola de texto

Abrimos una consola de texto local y escribimos:

```
ssh  UserNameFrontend@HostNameFrontend
```

Este comando nos pedirá la contraseña del usuario del Frontend. Si la autenticación fue exitosa, nos encontraremos en una sesión SSH remota, es decir los comandos que ejecutemos en esa consola se ejecutarán en el Frontend.



3 Ejecución

La ejecución de aplicaciones sobre el cluster de cómputo NO debe hacerse directamente sino a través del gestor de recursos.

Por cada ejecución, debe crearse un ***script*** para indicarle al gestor de recursos como ejecutar.

Cabe destacar, que la ejecución de la aplicación no es interactiva como en una ejecución tradicional. Una vez que finaliza la ejecución de la aplicación, el gestor de recursos dejará dos archivos: uno con la salida de la aplicación y otro con la salida de errores (si es que los hubo). Es importante indicarle al gestor de recursos el nombre y la ubicación de estos archivos.

3.1 Características generales de la ejecución

Las características del ***script*** dependerá de la partición elegida (Cluster Multicore, Intel Manycore o Cluster MultiGPU) y del tipo de ejecución (Secuencial o Paralela). Para cada caso se describirá el ***script*** específico pero, a modo de ejemplo, presentamos un modelo general y describimos los parámetros comunes.

3.1.1 Modelo de script

La creación del ***script*** consiste en crear un archivo con el siguiente contenido:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH –-exclusive
#SBATCH --partition=Blade
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
./miAplicación
```

La linea **`#!/bin/bash`** es el encabezado del script.

La línea **`#SBATCH -N 1`** le indica al gestor de recursos que debe reservar **un nodo** del cluster de cómputo y la línea **`#SBATCH --exclusive`** le indica que lo haga de forma exclusiva. Sin la línea de exclusividad, dos aplicaciones de dos usuarios diferentes podrían ejecutar en el mismo nodo al mismo tiempo interrumpiéndose.

La linea **`#SBATCH –partition=Blade`** indica la partición donde ejecutar. Cada partición tiene un nombre específico. Debemos poner **`Blade`** si queremos

ejecutar sobre el Cluster Multicore, **`XeonPHI`** si queremos ejecutar sobre el Intel Manycore y **`GPUS`** si queremos ejecutar sobre el Cluster MultiGPU. Si omitimos esta linea, el gestor de recursos tomará **Blade como la partición por defecto** y se enviará a ejecutar sobre el Cluster Multicore.

Las líneas **`#SBATCH -o directorioSalida/output.txt y #SBATCH -e directorioSalida/errors.txt`** indican el directorio y archivo de salida y de errores, respectivamente. Es importante que el directorio de salida exista sino retornará un error.

La última línea, **`./miAplicación`**, es el nombre de la aplicación a ejecutar. La linea comienza con `./` considerando que la aplicación se encuentra en el mismo directorio que el script. De no ser así, debe especificarse el directorio donde se encuentra la aplicación.

3.1.2 Ejecución del script

Suponemos que llamamos **`miScript.sh`** a nuestro script.

En primer lugar, debemos darle permisos de ejecución. Para esto ejecutamos el siguiente comando:

```
chmod +x miScript.sh
```

Luego, ejecutamos nuestro script mediante el comando:

```
sbatch ./miScript.sh
```

Si la aplicación recibe parámetros es posible pasarlos a través del script:

```
sbatch ./miScript.sh par1 par2
```

dentro del script podemos referirnos a estos parámetros como:

```
./miAplicación $1 $2
```

Donde `$1` será el primer parámetro pasado al script (`par1`), `$2` será el segundo (`par2`) y así siguiendo.

Si hay recursos disponibles, el gestor de recursos ejecutará la aplicación. Si no hay recursos disponibles, la ejecución quedará pendiente y el requerimiento se encolará en una cola de trabajos pendientes.

3.1.3 Control de aplicaciones

Podemos utilizar distintos comandos provistos por el gestor de recursos para, entre otras cosas, ver en que estado se encuentra nuestra aplicación (monitorización) o cancelarla.

Monitorización

Para ver en que estado se encuentra la aplicación enviada al gestor de recursos, utilizamos el siguiente el comando:

```
squeue
```

Este comando retorna una salida similar a:

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
297	colaA	app1	usuario1	PD	0:00	1	(Resources)
298	colaB	app2	usuario2	PD	0:00	1	(Priority)
295	colaB	appMPI	usuario2	R	0:01	1	nodo1
296	colaA	appOMP	usuario1	R	0:01	1	nodo2

Donde:

JOBID: identificador único del trabajo encolado.
PARTITION: el gestor de recursos puede manejar varias colas o particiones. Este campo indica en que cola fue encolado el trabajo.
USER: nombre del usuario propietario del trabajo.
ST: estado del trabajo. (R: running, PD: pending)
TIME: el tiempo que lleva de espera
NODES: el número de nodos solicitados
NODELIST (REASON): lista de nodos alocados

Cancelación de ejecuciones

Podemos cancelar la ejecución de una aplicación o desencolar un trabajo encolado mediante el comando:

```
scancel JOBID
```

3.2 Ejecución sobre el Cluster Multicore

Describimos la ejecución de aplicaciones secuenciales, aplicaciones paralelas con el modelo de memoria compartida (Pthreads y OpenMP), aplicaciones con el modelo de memoria distribuida (MPI) y aplicaciones híbridas (MPI-OpenMP y MPI-Pthreads)

El código fuente de las aplicaciones que se ejecutarán sobre el Cluster Multicore debe compilarse en el Frontend. Debe utilizarse el compilador GNU (gcc) para compilar aplicaciones secuenciales, Pthreads y OpenMP, mientras que las aplicaciones MPI e Híbridos deben compilarse con el compilador de OpenMPI (mpicc).

3.2.1 Aplicaciones secuenciales

Crear un script con el siguiente contenido:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errores.txt
./miAplicación
```

3.2.2 Aplicaciones paralelas de memoria compartida

Para el caso de aplicaciones **Pthreads** el gestor de recursos no requiere ninguna acción especial. Es responsabilidad del programador ejecutar la aplicación indicando, mediante un parámetro de entrada, el número de hilos a crear.

Se debe crear un script:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
./miAplicación nroHilos
```

Como se observa, uno de los parámetros de de la aplicación debe ser el número de hilos (**nroHilos**) con el que se quiera ejecutar. Luego, el programador debe crear ese número de hilos en el código.

Para el caso de **OpenMP** la creación de los hilos es implícita. Es el runtime system de OpenMP quien crea los hilos. El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
export OMP_NUM_THREADS=8
./miAplicacion
```

Sólo se debe agregar la línea **`export OMP_NUM_THREADS=8`** donde se indica el número de hilos con los que se quiere ejecutar.

3.2.3 Aplicaciones paralelas de memoria distribuida

Si queremos ejecutar una aplicación MPI tendremos varias alternativas:

- Ejecución sobre un único nodo
- Ejecución sobre varios nodos

Ejecución sobre un único nodo

Nos referimos a este tipo de ejecución cuando queremos ejecutar la aplicación MPI sobre un único nodo del Cluster Multicore (**nodo1 o nodo2, … o nodoX**).

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --tasks-per-node=8
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
mpirun  miAplicacionMPI
```

En este caso, se agrega la linea **`#SBATCH –tasks-per-node=8`** que indica al gestor de recursos que ejecute la aplicación en un único nodo utilizando 8 procesos.

Ejecución sobre varios nodos

Nos referimos a este tipo de ejecución cuando queremos ejecutar la aplicación MPI sobre varios nodos del Cluster Multicore (**nodo1 y nodo2, … y nodoX**). Por ejemplo, corremos 8 procesos sobre dos nodos, 4 en cada nodo.

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 2
#SBATCH --exclusive
#SBATCH --tasks-per-node=4
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
mpirun miAplicacionMPI
```

En este caso, la línea **`#SBATCH -N 2`** indica que se deben reservar 2 nodos. La línea **`#SBATCH --tasks-per-node=4`** indica que deben ejecutarse 4 procesos por nodo.

3.2.4 Aplicaciones paralelas híbridas

Nos referimos a aplicaciones Híbridas cuando combinamos dos modelos de programación:

- Modelo de memoria distribuida MPI - Modelo de memoria compartida OpenMP.
- Modelo de memoria distribuida MPI - Modelo de memoria compartida Pthreads.

Vamos a ejecutar un proceso por cada nodo y cada proceso creará el número de hilos que requiera para ejecutar.

Ejecución MPI-OpenMP

La creación de los hilos es implícita. Por ejemplo, corremos 2 procesos sobre dos nodos, uno en cada nodo. Luego, cada proceso creará 4 hilos.

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 2
#SBATCH --exclusive
#SBATCH --tasks-per-node=1
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
export OMP_NUM_THREADS=4
mpirun --bind-to none miAplicacionMPI
```

Es importante agregar el parámetro **`--bind-to none`**. Este parámetro resuelve un problema del Runtime System de OpenMPI evitando que todos los hilos se ejecuten sobre el mismo core.

Ejecución MPI-Pthreads

La creación de los hilos es explícita. Por ejemplo, corremos 2 procesos sobre dos nodos, uno en cada nodo.

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 2
#SBATCH --exclusive
#SBATCH --tasks-per-node=1
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
mpirun --bind-to none miAplicacionMPI nroHilos
```

Uno de los parámetros de de la aplicación MPI debe ser el número de hilos (**nroHilos**) con el que se quiera ejecutar. Luego, el programador debe crear ese número de hilos en el código.

3.3 Ejecución sobre el Intel Manycore

Describimos la ejecución de aplicaciones secuenciales, aplicaciones paralelas con el modelo de memoria compartida (Pthreads y OpenMP) y aplicaciones con el modelo de memoria distribuida (MPI).

El código fuente de las aplicaciones que correrán sobre el Intel XeonPHI KNL debe compilarse sobre esta arquitectura y utilizando los compiladores de Intel (OneAPI).
Usaremos el compilador mpiicc para compilar aplicaciones MPI y el compilador icc para compilar aplicaciones secuenciales, Pthreads y OpenMP.
Como accederemos al XeonPHI a través del gestor de recursos, debemos incluir dentro del script las líneas de compilación.
Además, es necesario que el script incluya la carga de las variables de entorno provistas por OneAPI:

```
source $ONEAPI_PATH/setvars.sh > /dev/null 2>&1
```

3.3.1 Aplicaciones secuenciales

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --partition=XeonPHI
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
source $ONEAPI_PATH/setvars.sh > /dev/null 2>&1
icc -o miAplicacion miAplicacion.c
./miAplicacion
```

3.3.2 Aplicaciones paralelas de memoria compartida

Para el caso de aplicaciones **Pthreads** el gestor de recursos no requiere ninguna acción especial. Es responsabilidad del programador ejecutar la aplicación indicando, mediante un parámetro de entrada, el número de hilos a crear.

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --partition=XeonPHI
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
source $ONEAPI_PATH/setvars.sh > /dev/null 2>&1
icc -pthread -o miAplicacion miAplicacion.c
./miAplicacion nroHilos
```

Como se observa, uno de los parámetros de de la aplicación debe ser el número de hilos (**nroHilos**) con el que se quiera ejecutar. Luego, el programador debe crear ese número de hilos en el código.

Para el caso de **OpenMP** la creación de los hilos es implícita. Es el runtime system de OpenMP quien crea los hilos. El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --partition=XeonPHI
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
source $ONEAPI_PATH/setvars.sh > /dev/null 2>&1
icc -fopenmp -o miAplicacion miAplicacion.c
export OMP_NUM_THREADS=64
./miAplicacion
```

Sólo se debe agregar la línea **`export OMP_NUM_THREADS=64`** donde se indica el número de hilos con los que se quiere ejecutar.

3.3.3 Aplicaciones paralelas de memoria distribuida

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --partition=XeonPHI
#SBATCH --tasks-per-node=64
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
source $ONEAPI_PATH/setvars.sh > /dev/null 2>&1
mpiicc -o miAplicacion miAplicacion.c
mpirun ./miAplicacion
```

En este caso, se agrega la linea **#SBATCH –tasks-per-node=64** que indica al gestor de recursos que ejecute la aplicación en un único nodo utilizando 64 procesos.
Tener en cuenta que el compilador es mpiicc. Además, considerar que si la aplicación está en el mismo directorio que el script debe agregarse ./ ,sino falla.

3.4 Ejecución sobre el Cluster MultiGPU

Describimos la ejecución de aplicaciones paralelas que se ejecutan sobre una GPU o las dos GPUs de un nodo del Cluster MultiGPU.

El código fuente de las aplicaciones que se ejecutarán sobre alguno de los nodos del Cluster MultiGPU debe compilarse en el Frontend. Para ello, utilizamos el compilador de Nvidia CUDA (nvcc) de la siguiente forma:

```
nvcc -o miAplicacion miAplicacion.cu
```

El script debe tener la siguiente forma:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --partition=GPUS
#SBATCH -o directorioSalida/output.txt
#SBATCH -e directorioSalida/errors.txt
./miAplicacion
```

En caso de querer utilizar las dos GPUs de un nodo, el código debe interactuar con OpenMP de manera de crear un hilo para la gestión de cada GPU disponible. El número de hilos a crear se indica en una directiva OpenMP y puede manejarse automáticamente en el código fuente en función de las GPUs disponibles. Sin embargo, es necesario compilar la aplicación con la siguiente línea:

```
nvcc -Xcompile -fopenmp -o miAplicacion miAplicacion.cu
```

El script no requiere de cambios y será igual al script anterior.

4 Resultados

Los resultados de cada ejecución quedarán en los archivos de salida que le indiquemos al gestor de recursos. Para recuperarlos del cluster sólo tenemos que utilizar el protocolo SCP de manera inversa.