# LABORATORIO NACIONAL DE SUPERCÓMPUTO DEL SURESTE DE MÉXICO

# Guía básica de utilización de la Supercomputadora *Cuetlaxcoapan*

**Aprobado por:** Dr. Manuel Martín Ortiz

**Fecha de Publicación:** 12/02/2016

**1ra. Edición**

## Control de cambios

| No. de Edición | Fecha | Motivo de la edición |
|---|---|---|
| 1ra. Edición | 12/02/2016 | Primer Ejemplar |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Elaborado por: Nombre y firma del Administrador de Proyectos

# INDICE DE CONTENIDO

# 1. Introducción

Este documento presenta información básica para la conexión a su cuenta de usuario y el uso de un intérprete de comandos, así como un conjunto de instrucciones y procedimientos para ejecutar de manera apropiada sus tareas de cálculo en la Supercomputadora *Cuetlaxcoapan* del Laboratorio Nacional de Supercómputo (LNS) del Sureste de México.

# 2. Distribución General de los Recursos de Cómputo del LNS

Es importante que el usuario del LNS conozca la forma en que trabajará en la supercomputadora *Cuetlaxcoapan*, por lo que le sugerimos poner especial atención a la siguiente información:

- Deberá seguir los procedimientos descritos en la siguiente guía para realizar las tareas habituales de uso de la supercomputadora, desde establecer la conexión, el envío de tareas de cálculo, hasta la transferencia de archivos entre su computadora local y la supercomputadora.
- Por motivos de eficiencia y seguridad, existe una estructura establecida en la supercomputadora (véase la Fig. 1) en la que se delimita claramente el espacio en el que trabajará el usuario y los recursos administrados por el gestor SLURM.

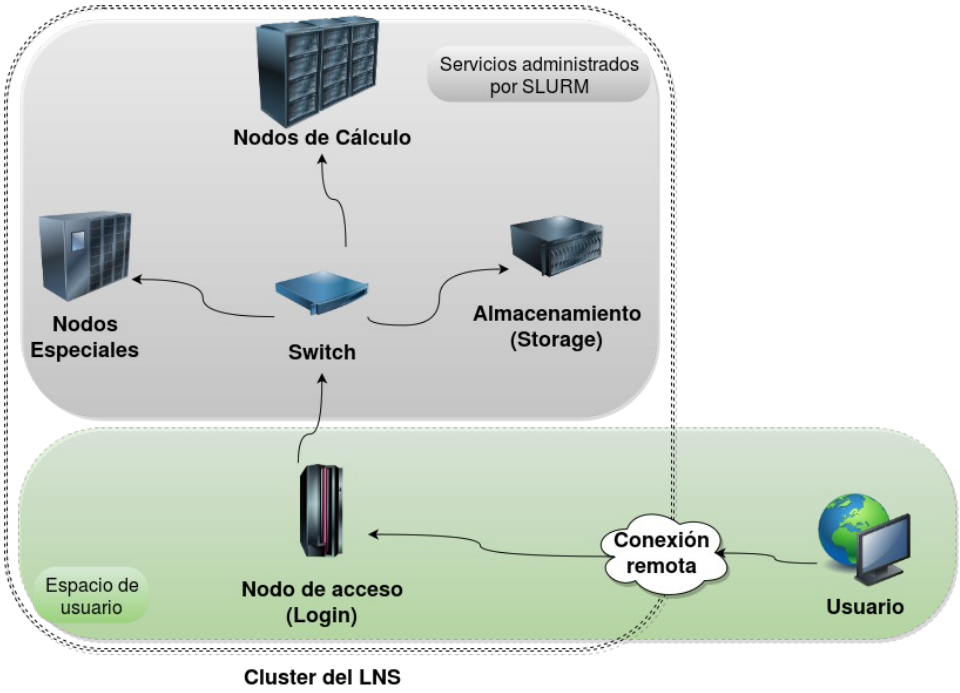


Figura 1: Diagrama de acceso al cluster de supercómputo del LNS.

- Para que los recursos sean utilizados de la manera más eficiente, es de gran importancia que respete el espacio de trabajo designado al usuario y que haga uso de la supercomputadora a través del sistema SLURM, ya que esto permitará

balancear adecuadamente los recursos compartidos entre sus tareas y las de otros usuarios.

- La supercomputadora C*uetlaxcoapan* está compuesta de 136 nodos estándares de cálculo, 6 nodos especiales y un sistema de almacenamiento de 520 TB, interconectados en una red Infiniband con topología de estrella mediante un switch central (véase la Fig. 1). Estos recursos son administrados por el sistema SLURM (de las siglas en inglés de *Simple Linux Utility for Resource Management*). El acceso a los recursos se lleva a cabo mediante un nodo de gestión (login) que constituye el entorno donde los usuarios deberán realizar sus labores habituales de preparación, envío y monitoreo de tareas de cálculo, así como de recuperación de los datos generados.
- Es importante aclarar que no se permite realizar labores de postprocesamiento de datos en el nodo de login. El usuario deberá transferir los datos generados a su computadora local para postprocesamiento (visualización, etc.)
- Todos los usuarios tienen la misma prioridad de acceso a los recursos (nodos de cálculo y almacenamiento).
- Cada usuario tiene asignado un espacio de almacenamiento personal (directorio `HOME=/home/cuenta`) donde puede almacenar hasta 50 GB de información. Este espacio de almacenamiento es global (visible en todos los nodos) y esta destinado únicamente para scripts de cálculo, aplicaciones desarrolladas por el usuario y aplicaciones comerciales de uso personal. No se permite guardar datos de cálculos u otro tipo de datos en el directorio `HOME` (consultar las políticas de uso del LNS en www.lns.buap.mx).
- Los datos generados por el usuario deberán almacenarse en el directorio `SCRATCH=/scratch/cuenta`). Este directorio también es global, es decir, visible por todos los nodos de cálculo. Es recomendable respaldar periódicamente la información almacenada en `SCRATCH` ya que el LNS no se hace responsable de la integridad de la misma ante posibles fallas de hardware y software (consultar las políticas de uso del LNS en www.lns.buap.mx).
- El tiempo de cálculo se mide en *horas-core*. Cada nodo estándar de cálculo posee 24 cores (2 sockets Intel Xeon E5-2680 v3 a 2.5 Ghz con 12 cores/socket) y 128 GB de memoria RAM compartida.
- Cada usuario tiene reservados por default 240 cores de cálculo que se pueden utilizar simultáneamente si es necesario. Estos cores no están ligados a algún nodo específico, es decir, se pueden utilizar de manera indistinta en cualquier nodo de cálculo. Sin embargo, es importante utilizar afinidad entre cores cuando se envían tareas paralelas (véase la sección de SLURM más adelante).
- El usuario debe gestionar personalmente su tiempo de cálculo. Es decir, debe decidir adecuadamente, en base a criterios de eficiencia, cuantos cores deberá utilizar en cada tarea. Hay que tener en cuenta, por ejemplo, que usando los 240 cores simultáneamente en un mismo cálculo implica gastar 240 horas-core por cada hora de tiempo transcurrido. De esta forma, si su job corre durante un día completo (24 horas) gastará 5760 horas-core por día. Es decir, si su proyecto fué catalogado como de inicio con 50,000 horas-core de cálculo, usando los 240 cores

simultáneamente en cada tarea de cálculo se gastaría las 50,000 horas-core en menos de 10 días.

# 3. Guía básica de usuario

El conocimiento de la línea de comandos de Linux es de importancia fundamental para la utilización de la supercomputadora *Cuetlaxcoapan*, tanto para establecer una conexión y acceder a su cuenta, como para enviar tareas de cálculo y manipular de forma remota los datos generados.

Los sistemas operativos Linux y Mac OS X disponen por defecto de una aplicación conocida como *terminal* (véase la Fig. 2) que abre una sesión en el sistema usando un intérprete de comandos en modo de texto (también conocido como *shell*). La terminal muestra una ventana con el símbolo $ o > seguido de un prompt como _ o ▉ que generalmente parpadea para indicarnos que el intérprete está listo para recibir órdenes. El intérprete de comandos constituye entonces la manera más sencilla de interaccionar con la computadora, sobre todo cuando esta se utiliza remotamente. El nodo de login de la supercomputadora *Cuetlaxcoapan,* utiliza RedHat Enterprise Linux versión 6.6 y el intérprete de comandos por defecto es bash (www.gnu.org/software/bash). Por lo tanto, todos los comandos que describiremos de aquí en adelante serán ejecutados en bash.

Figura 2: La terminal de comandos de Linux o Mac OS X tiene un ícono similar a éste.

Incluso aunque su computadora de escritorio utilice el sistema operativo Windows (véase *Conexión SSH desde Windows*) también requiere familiarizarse con el intérprete de comandos de Linux, ya que al conectarse a la supercomputadora *Cuetlaxcoapan* estará utilizando el sistema operativo Linux.

Bash es un intérprete que dispone de la capacidad extremadamente útil de *autocompletado* de comandos. Utilizando esta capacidad, puede reconocer comandos escribiendo solo las primeras letras y solicitando al intérprete completar el comando presionando la tecla `Tab`. Si esta tecla se presiona más de una vez, bash muestra distintas opciones de completar el comando, o rutas de archivos y directorios y palabras reservadas por el sistema que cumplen con el indicio que escribió, e incluso autocompleta la línea entera si solo existe una opción disponible. Sugerimos que pruebe esta opción y la utilice con regularidad.

## 3.1. Estructura básica de un comando del shell

Tradicionalmente, la estructura de un comando de bash o de algún otro shell de Linux como ksh, tcsh, etc., se asemeja a la siguiente:

```
comando <opciones> <parámetro1> <parámetro2> ...
```

donde las opciones suelen colocarse mediante letras precedidas por un signo de guión corto (-). Los parámetros son argumentos que el comando va a utilizar para desarrollar su función. Por ejemplo, el comando `ls` sin opciones ni argumentos presenta una lista corta de los archivos de un directorio incluyendo subdirectorios, mientras que `ls -l` presenta

la misma información pero ordenada en forma de lista incluyendo los atributos (tamaño, permisos, etc.) de los archivos y subdirectorios. Si desea información más detallada de un comando específico siempre podrá obtenerla directamente desde la terminal, ejecutando la instrucción `man comando` o `comando --help`.

A continuación se presentan algunos comando útiles de Linux junto con una descripción simple de su función y un ejemplo de uso.

## 3.2. Comandos de gestión de ficheros y directorios

Al establecer una sesión en Linux mediante la línea de comandos el usuario será posicionado por default en el directorio `/home/cuenta`. El usuario posee privilegios para manipular información en este directorio, es decir, crear, modificar y borrar archivos y subdirectorios. La tabla 1 muestra algunos comandos que son básicos para navegar por el sistema de archivos y directorios, y con ello operar eficientemente en Linux. La figura 3 muestra también ejemplos de ejecución de éstos.

Tabla 1: Comandos básicos de gestión de ficheros y directorios

| Comando | Descripción | Ejemplo |
|---|---|---|
| `cat arch` | Muestra el contenido del archivo `arch` | `cat programa.f` |
| `cd dir` | Cambiar del directorio actual a `dir` | `cd programas` |
| `ls dir` | Lista el contenido del directorio `dir` | `ls programas` |
| `rm arch` | Elimina el archivo `arch` | `rm programa.c` |
| `cp arch1 arch2` | Copia el archivo `arch1` a `arch2` | `cp programa.c programa1.c` |
| `pwd` | Muestra la ruta del directorio actual | `pwd` |
| `mkdir dir` | Crea un nuevo directorio | `mkdir resultados` |
| `rmdir dir` | Elimina directorios | `rmdir pruebas` |
| `man command` | Despliega el manual de uso de un comando | `man ls` |
| `touch arch` | Crea un archivo vacío | `touch programas/hola.c` |
| `mv arch dir` | Mueve el archivo `arch` al directorio `dir` | `mv hola.c programas` |
| `clear` | Limpia la terminal | `clear` |
| `du -h dir` | Reporta el tamaño del directorio `dir` | `du -h` |
| `grep patron arch` | Busca un `patrón` o expresión en un archivo `arch` | `grep OPEN programa.f` |

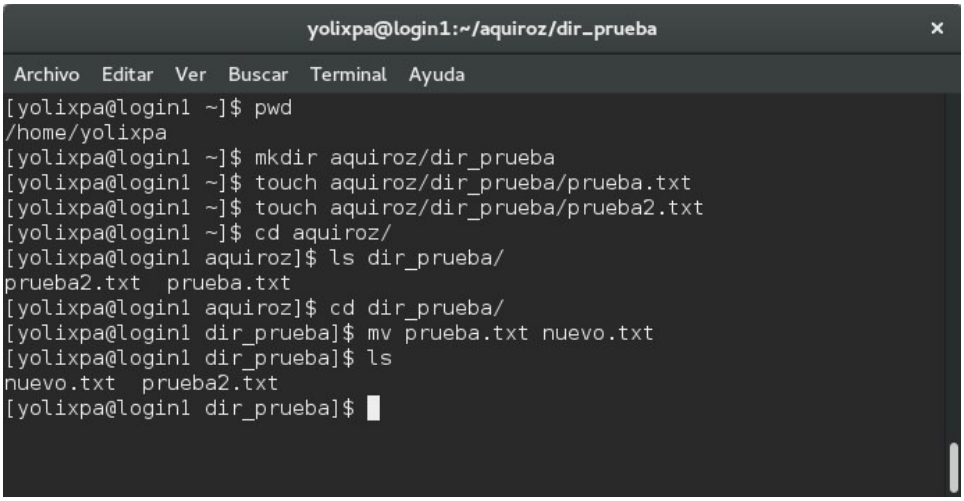


Figura 3: Ejemplos de ejecución de comandos de gestión de ficheros y directorios

## 3.3. Método de acceso a su cuenta en la supercomputadora

A continuación se presenta el procedimiento para establecer una conexión desde su computadora local a su cuenta en la supercomputadora *Cuetlaxcoapan* utilizando SSH.

Por motivos de seguridad, el acceso se lleva a cabo utilizando el servidor `ui.fcfm.buap.mx` (con dirección IP `148.228.128.223`) como nodo de acceso intermedio. Ud. tiene una cuenta en este servidor con el mismo nombre que en la supercomputadora *Cuetlaxcoapan* e inicialmente con el mismo password.

### 3.3.1. Conexión SSH desde Linux o Mac OS X

SSH (Secure SHell) es un protocolo para establecer una sesión remota en una computadora en forma segura usando la línea de comandos. Es la única opción permitida para accesar la supercomputadora *Cuetlaxcoapan*.

El acceso a la supercomputadora desde su computadora local usando Linux o Mac OS X se realiza de manera sencilla abriendo primero la terminal y conectándose luego a `ui.fcfm.buap.mx` (con dirección IP `148.228.128.223`) mediante la orden:

```
ssh cuenta@ui.fcfm.buap.mx
```

o bien

```
ssh cuenta@148.228.128.223
```

Al establecer la conexión es posible que `ssh` genere primero una clave de la maquina remota (`ui.fcfm.buap.mx` o `148.228.128.223`) y le solicite aceptarla tecleando `yes` en la terminal. A continuación, ssh solicita el password de la cuenta:

```
cuenta@ui.fcfm.buap.mx's password:
```

o bien

```
cuenta@148.228.128.223's password:
```

Al introducir el password correctamente obtenemos la linea de comandos en nuestra `cuenta` en `ui.fcfm.buap.mx`:

```
Last login: Fri Feb 12 00:00:00 2016 from 195.221.47.54

[cuenta@ui ~]$
```

La Fig. 4 muestra un ejemplo de conexión a `ui.fcfm.buap.mx` desde la computadora local de un usuario corriendo Ubuntu Linux:

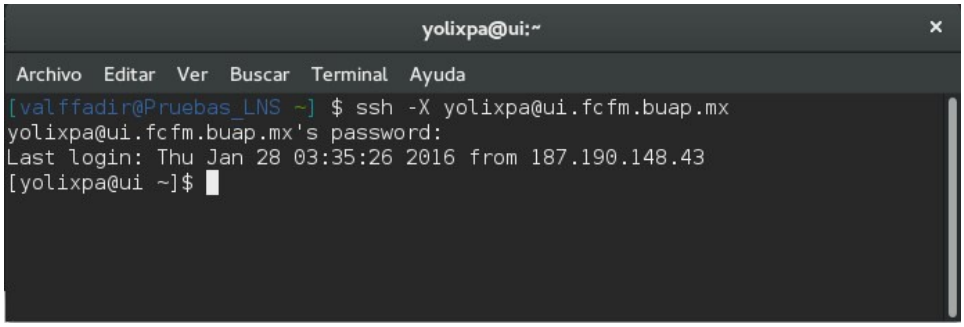


Figura 4: Ejemplo de conexión a `ui.fcfm.buap.mx` desde una terminal Linux

Es importante cambiar inmediatamente el password de su cuenta en `ui.fcfm.buap.mx` utilizando la orden:

```
yppasswd
```

Desde el servidor `ui.fcfm.buap.mx` deberá conectarse al nodo de acceso (login) de la supercomputadora *Cuetlaxcoapan* mediante la orden:

```
ssh cuenta@192.168.170.213
```

Al igual que en el caso de la conexión a `ui.fcfm.buap.mx`, ssh generará la primera vez una clave para `192.168.170.213` y le solicitará aceptarla tecleando `yes` en la pantalla. A continuación le pedirá el password en `192.168.170.213`:

```
cuenta@192.168.170.213's password:
```

Al introducir el password correctamente aparecerá la línea de comando de bash en el nodo de login de la supercomputadora Cuetlaxcoapan:

```
Last login: Fri Feb 12 00:00:00 2016 from ui.fcfm.buap.mx
****************************************
*                                      *
*   Bienvenidos a login1.lns.buap.mx!  *
*                                      *
****************************************
cuenta@login1[~]$
```

A continuación podemos empezar a utilizar la supercomputadora.

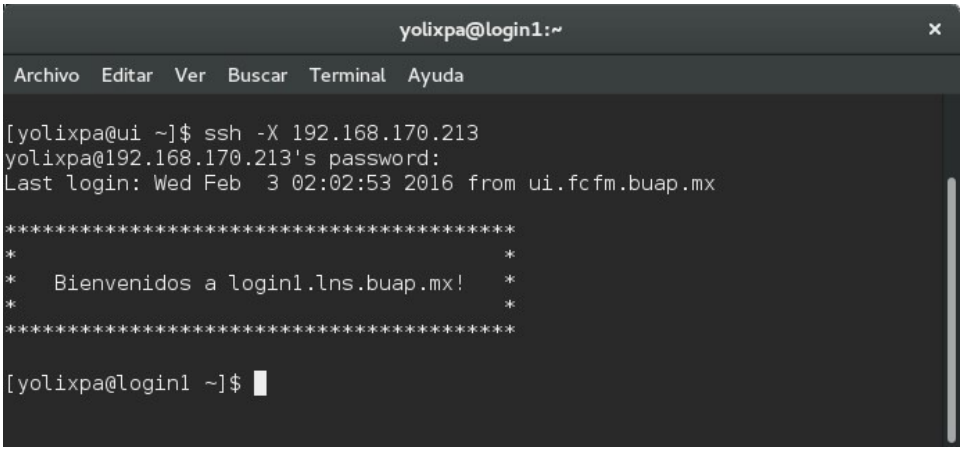


**Figura 5: Ejemplo de conexión a la supercomputadora *Cuetlaxcoapan* después de acceder a `ui.fcfm.buap.mx`.**

## Conexión SSH desde Windows

El sistema operativo Windows no posee una línea de comandos compatible con Linux. Por esta razón, debemos utilizar una aplicación especial para establecer una conexión SSH hacia una maquina Linux. Por simplicidad, le recomendamos utiizar PuTTY, que es una aplicación libre disponible en el sitio web: http://www.putty.org/. Otra aplicación útil es Bitvise SSH Client (Tunnelier), que es

una aplicación comercial que se distribuye libremente y esta disponible en el sitio web https://www.bitvise.com/ssh-client-download. Bitvise SSH Client provee también una interface gráfica para transferir archivos via SSH, por lo que es muy recomendable instalarla.

Al correr PuTTY nos aparece primero una interface para configurar la conexión (ver Fig. 6). En principio, solo debemos introducir el nombre (Host Name) o la dirección IP de la máquina remota para establecer la conexión, como aparece en la Fig. 6. Al presionar la tecla Open de la ventana de configuracion nos aparece la terminal de comandos de PuTTY, donde nos solicita el nombre de la cuenta y password para abrir una sesión en la máquina remota. En la Fig. 7 se muestra un ejemplo de conexión a `ui.fcfm.buap.mx` utilizando PuTTY.

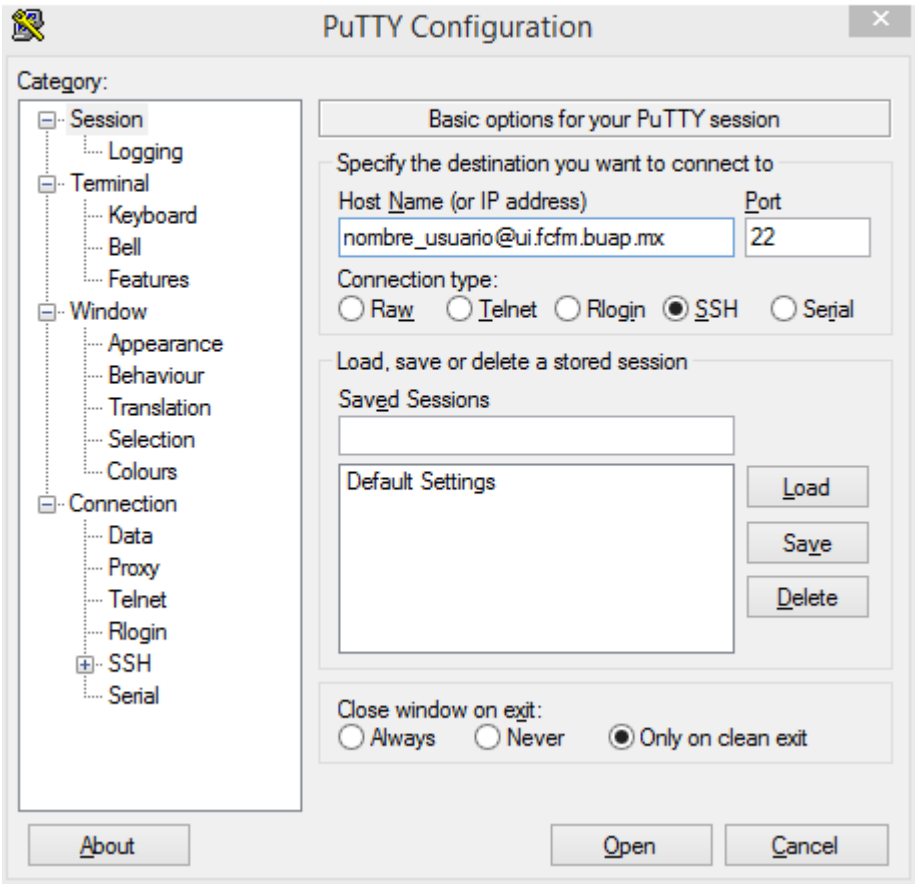


Figura 6: Ventana de onfiguración de PuTTY para una conexión SSH


Figura 7: Ejemplo de conexión a `ui.fcfm.buap.mx` usando PuTTY

A partir de la sesión de PuTTY o Bitvise SSH Client en `ui.fcfm.buap.mx` es posible establecer la conexión al nodo de login de la supercomputadora Cuetlaxcoapan de la misma forma que se hizo para Linux o Mac OS X, es decir, escribiendo:

```
ssh cuenta@192.168.170.213
```

y tecleando `yes` cuando SSH nos pide aceptar la clave de la maquina remota, para introducir a continuacion el password de la cuenta y obtener la línea de comandos de bash en nuestra cuenta en la supercomputadora.

### 3.3.2. Procedimiento posterior a la primera conexión con su cuenta en el LNS

Es muy recomendable que, después de realizar la primera conexión SSH a su cuenta la supercomputadora *Cuetlaxcoapan*, realice lo siguiente:

- Cambie la contraseña con la que accedió a su cuenta por una más segura. Recomendamos que utilice un mínimo de 8 caracteres y que incluya una combinación de letras, números y símbolos (!"#$%&/()=). Para ello, ejecute el comando:

  ```
  yppasswd
  ```

  y enseguida aparecerán en pantalla las opciones para ingresar su viejo password, el nuevo y la confirmación del mismo.

- Genere las claves de SSH de los nodos de cálculo para activar el acceso sin password desde el nodo de login. Esto se lleva a cabo con los siguientes comandos:

1. Borrar la carpeta `.ssh` de su cuenta:

```
rm -rf $HOME/.ssh
```

2. Crear de nuevo la carpeta `.ssh` en `HOME`:

```
mkdir $HOME/.ssh
```

3. Cambiar los permisos de `.ssh` para que sólo el propietario tenga acceso, mediante el comando:

```
chmod 700 $HOME/.ssh
```

4. Generar la clave SSH personal con el comando:

```
ssh-keygen
```

El comando presenta varias preguntas, Ud. solo deberá oprimir la tecla `Enter` en cada una de ellas para aceptar los valores por defecto.

5. A continuación, entrar al directorio .ssh :

```
cd $HOME/.ssh
```

6. Copiar la clave pública de usuario (`id_rsa.pub`) al archivo de claves autorizadas (`authorized_keys`) :

```
cp id_rsa.pub authorized_keys
```

7. Volver a `HOME`:

```
cd $HOME
```

8. Por último, ejecutar los siguientes scripts en la terminal para generar las claves de los nodos:

```
/software/ricardo/claves_ssh_nombre_largo.sh
/software/ricardo/claves_ssh_nombre_corto.sh
```

Deberá aceptar la clave generada para cada nodo escribiendo la palabra `yes` cada vez que le sea solicitado.

## 3.4. Transferencias de archivos

Es importante tener en cuenta que al utilizar una conexión intermedia a `ui.fcfm.buap.mx` la transferencia de archivos desde su computadora local a la supercomputadora *Cuetlaxcoapan* y viceversa no se puede realizar directamente.

### 3.4.1. Procedimiento para transferencia desde su computadora local hacia la supercómputadora

1. En su computadora local debe transferir primero sus archivos a su cuenta en `ui.fcfm.buap.mx` utilizando los comandos de transferencia de archivos que se mencionan en la sección *Comandos para transferencia de archivos en la sección 3.4.3*.
2. Conectarse a su cuenta en `ui.fcfm.buap.mx` usando SSH, tal como se ha descrito en la sección 3.3.
3. Transferir los archivos a su cuenta en `192.168.170.213` (nodo de login de la supercomputadora) usando los comandos que se mencionan en la sección *Comandos para transferencia de archivos en la sección 3.4.3*.

### 3.4.2. Procedimiento para transferencia desde la supercomputadora hacia su computadora local

1. Conectarse a su cuenta en la supercomputadora usando SSH, tal como se ha descrito en la sección 3.3.
2. Transferir los archivos a su cuenta en `ui.fcfm.buap.mx` utilizando los comandos de transferencia de archivos que se mencionan en la sección *Comandos para transferencia de archivos en la sección 3.4.3*.
3. Transferir los archivos desde `ui.fcfm.buap.mx` a su computadora local usando los comandos de transferencia de archivos que se mencionan en la sección *Comandos para transferencia de archivos en la sección 3.4.3*.

### 3.4.3. Comandos para transferencia de archivos

A continuación se describen algunos comandos que puede utilizar para realizar transferencia de archivos.

#### 3.4.3.1. rsync

Es un comando utilizado para copiar y sincronizar archivos desde o hacia un sistema remoto; utiliza un sistema de compresión y descompresión de datos al momento de enviar o recibir archivos y, si ha realizado la copia de un archivo previamente, al actualizar su copia es capaz de transferir únicamente los cambios entre el archivo origen y destino; por lo anterior se considera la opción más rápida entre de los comandos de transferencia tradicionales, como `scp`. Recomendamos que utilice `rsync` antes que otros.

Antes de utilizar `rsync`, asegúrese de tenerlo instalado en su sistema. Para verificar esto puede simplemente escribir el comando

```
rsync --version
```

en su terminal; si el programa despliega el nombre y versión de `rsync` instalada, puede continuar.

La sintaxis de un comando `rsync` es la siguiente:

```
rsync [opciones] fuente destino
```

donde `fuente` es la ruta y nombre del archivo o directorio que quiere transferir; `destino` es la dirección de la computadora a la que desea acceder; y en `opciones` se establecen distintos modos de ejecución del comando rsync, siendo las más interesantes: `-v` que ofrece información detallada del proceso, `-a` que permite una copia de archivos y directorios junto con sus propiedades y rutas originales, y `-z` que habilita las opciones de compresión al transferir. Para combinar las 3 opciones al mismo tiempo, la sintaxis puede ser `-azv`.

Tanto la ruta de los archivos/directorios `fuente` como `destino` pueden ser en un sistema remoto, por lo que se utiliza el formato descrito en la sección `Procedimiento de transferencia de ficheros` para escribir las rutas de los archivos o directorios a los que desea acceder.

A continuación se ejemplifica el uso del comando `rsync` utilizando el procedimiento descrito en la sección anterior para la transferencia de un archivo desde su cuenta del cluster de supercómputo hacia su computadora.

1. Abra una ventana con el intérprete de comandos de su sistema (es probable que lo encuentre con el nombre de *terminal o consola*) (véase figura 1); y establezca una conexión SSH con su cuenta en el cluster intermedio Fénix (véase *Comandos para transferencia de archivos*).
2. Una vez que se conectó a su cuenta en el cluster intermedio, considere ésta como su cuenta local y copie el archivo que necesita desde su cuenta en el cluster del LNS (en su cuenta remota) mediante el comando:

   ```
   rsync -avz usuario@192.168.170.213:ruta/archivo.ext rutalocal/archivo.ext
   ```

   e ingrese la contraseña del usuario con el que intenta *loguearse* después de ejecutarlo. Recuerde agregar la ruta y nombre completos del archivo, incluyendo la extensión del mismo.

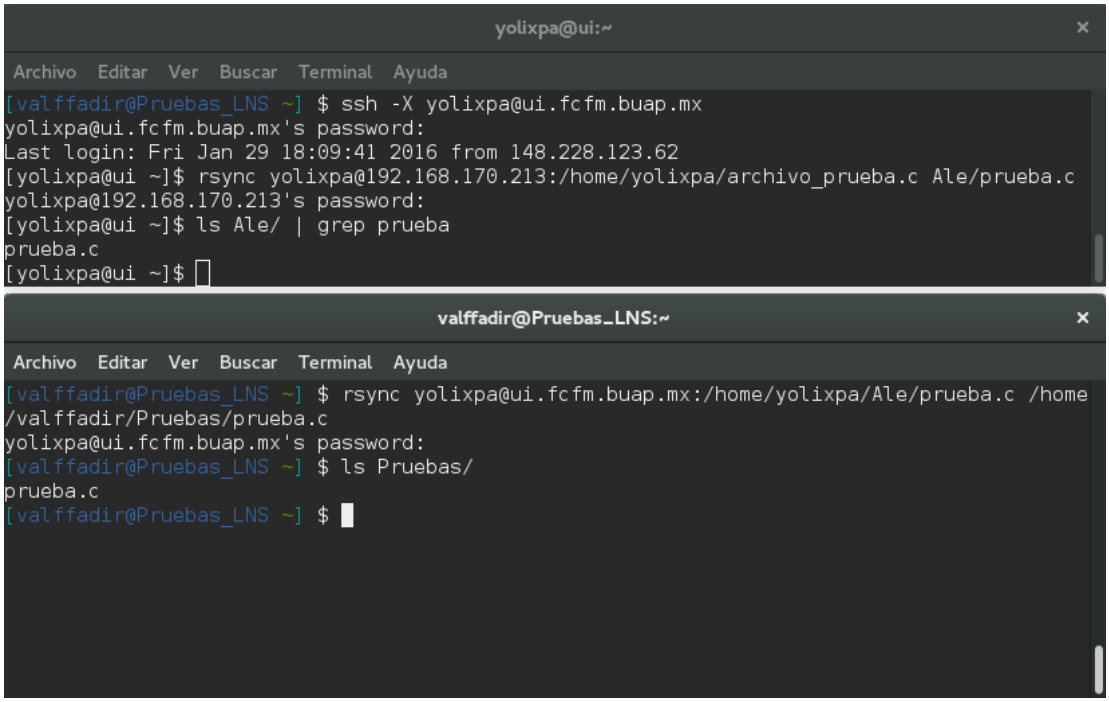


**Figura 8: Ejemplo de transferencia de archivos utilizando rsync. En la ventana superior se observa la transferencia desde el cluster de supercómputo hacia el cluster Fénix. En la inferior, la transferencia desde éste último hacia una computadora de esc**

3. Abra una nueva terminal en su computadora que es ahora la cuenta local, y transfiera el archivo del paso anterior desde su cuenta en el cluster Fénix, que ahora consideramos la cuenta remota, mediante el comando:

```
rsync -avz usuario@ui.fcfm.buap.mx:ruta/archivo.ext rutalocal/archivo.ext
```

e ingrese la contraseña del usuario con el que intenta *loguearse* después de ejecutarlo. Recuerde agregar la ruta y nombre completos del archivo, incluyendo la extensión del mismo.

## 3.5. Chequeo y carga de software disponible en el LNS

Por motivos de seguridad y rendimiento, antes ejecutar programas en el cluster de supercómputo requiere indicar qué herramientas de software va a utilizar cada que inicie sesión en su cuenta. Para ello, una vez establecida la conexión con su cuenta en el LNS, se deben cargar como módulos complementarios cualquiera de las herramientas y programas que el Laboratorio tiene a su disponibilidad.

Si utiliza los *scripts* proporcionados por ésta guía en la sección *Scripts para ejecución de programas en SLURM*, ya habrá cargado los módulos necesarios para ejecutar el programa que desea. No obstante, si requiere ejecutar algún otro programa que no se encuentra detallado en dicha sección, probablemente requiera cargar más módulos en su sesión de conexión.

Los comandos que necesitará para realizar dicha tarea son los siguientes:

- Para verificar los módulos disponibles para el usuario y el directorio que los contiene es:

```
module avail
```

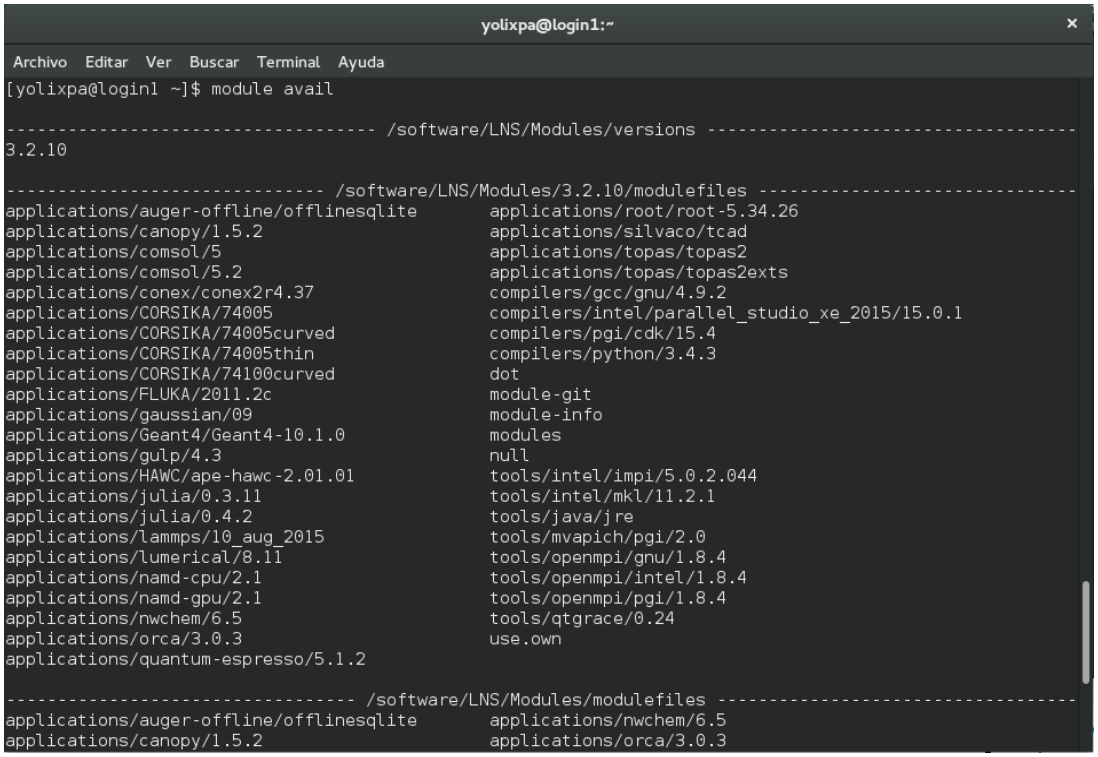


Figura 9: Ejemplo de ejecución del comando *module avail*

- Para cargar un módulo específico use:

```
module load [directorio]
```

donde `directorio` es la ruta y versión del módulo que desea cargar. La obtiene cuando ejecuta el comando `module avail`

- Para verificar los módulos que ya tiene cargados en su sesión:

```
module list
```

- Para volver a inhabilitar alguno de los módulos que ya tiene cargados en su sesión:

```
module unload [directorio]
```

donde `directorio` es la ruta y versión del módulo que desea deshabilitar.

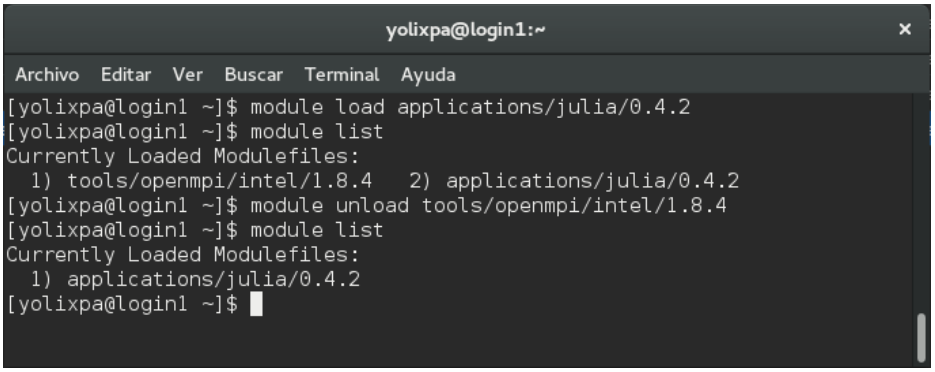


Figura 10: Ejemplos de ejecución de comandos de uso de módulos

## 3.6. Edición de archivos en la consola de comandos

Para realizar modificaciones menores en los archivos en su cuenta, existen diversas opciones de software a su disposición. Programas como `vi`, `vim` y `nano` son comunes y muy utilizados para realizar dichas ediciones de texto en archivos. Recomendamos nuevamente que busque más información sobre tales editores para sacar el máximo provecho a sus funciones.

A continuación se describen las generalidades de los editores antes nombrados:

### 3.6.1. vi

Es un editor de texto incluido en todos los sistemas Unix, es decir, tanto Linux como Mac Os. Para lanzarlo, utilice simplemente el comando:

```
vi [archivo]
```

donde `archivo` es el nombre (y la ruta si no se encuentra el archivo en el directorio de trabajo actual) que deseamos editar con `vi`.

Tiene dos modos de operación, el modo de *edición de texto* y el *modo de comandos*. El primero sirve, como su nombre lo indica, para ingresar datos de texto hacia el documento; el segundo por su parte permite la escritura de comandos cortos que editen de manera eficaz el contenido del texto en el archivo.

En la tabla 2 encontrará algunos de los comandos más utilizados en vi.

Existe una inmensa variedad de comandos disponibles para el editor *vi*. Reiteramos la utilidad de investigar más acerca de los mismos. En internet puede encontrar comandos avanzados de vi.

Tabla 2: Comandos básicos de editores vi y vim

| Modo de edición | Modo de comandos |
|---|---|

| | | | |
|---|---|---|---|
| Comando | Utilidad | Comando | Utilidad |
| `a, i` | inicia modo de edición (variantes) | `<ESC>` | Entra al modo de comandos |
| `o, O` | inicia edición insertando líneas | `:w nombre` | Escribe en el archivo "nombre" |
| `:r archivo` | inserta el contenido de un archivo | `:wq` | Escribe el archivo y sale de vi |
| | | `:q!` | Sale del editor sin guardar |

### 3.6.2. vim

Es un editor de texto sucesor de *vi* ya que incorpora características extras en la edición de texto, sobre todo para edición de código en varios de los lenguajes de programación más comunes. Se encuentra facilmente a través del manejador de paquetes de su sistema operativo Unix, es decir, tanto Linux como Mac Os pueden ocuparlo. Para lanzarlo, utilice simplemente el comando:

```
vim [archivo]
```

donde `archivo` es el nombre (y la ruta si no se encuentra el archivo en el directorio de trabajo actual) que deseamos editar con `vim`.

Al igual que `vi`, tiene dos modos de operación, el modo de *edición de texto* y el *modo de comandos*. El primero sirve, como su nombre lo indica, para ingresar datos de texto hacia el documento; el segundo por su parte permite la escritura de comandos cortos que editen de manera eficaz el contenido del texto en el archivo. Los comandos que utiliza en `vi`, también puede utilizarlos en `vim` (véase tabla 2). Puede encontrar comandos avanzados de vim fácilmente en la red.

### 3.6.3. nano

Es un editor de texto muy básico pero eficiente, utilizado en sistemas Unix. Para ocupar *nano* utilice el comando:

```
nano [archivo]
```

donde `archivo` es el nombre (y la ruta si no se encuentra el archivo en el directorio de trabajo actual) que deseamos editar con `nano`.

Éste editor de texto, a diferencia de `vi` *y* `vim`, no tiene un modo de comandos, pues en todo momento se encuentra en lo que denominamos en dichos editores como *modo de edición*. La forma en que se introducen los comandos es principalmente mediante la combinación de teclas `Ctrl+[letra/tecla(s)]`, siendo los comandos más utilizados los de copiado `(Ctrl+Shift+C)`, pegado `(Ctrl+Shift+V)` y cierre del editor `(Ctrl+X)`. En la parte inferior de la pantalla del editor puede encontrar otros de los comandos más utilizados. Si desea buscar aún más comandos para `nano`, sugerimos consultar ayuda en la red.

# 4. SLURM

## 4.1. ¿Qué es SLURM?

Como su nombre lo indica (Simple Linux Utility for Resource Management), SLURM es un administrador *open-source* de carga de trabajos diseñado para clusters Linux de cualquier dimensión. Es utilizado en varias de las computadoras más grandes del mundo.

Provee tres funciones claves:

1. Asigna a los usuarios un acceso exclusivo y/o no exclusivo a recursos (nodos computacionales) por un periodo de tempo determinado de tal forma que puedan realizar sus tareas.
2. Proporciona un marco para iniciar, ejecutar y monitorear el trabajo (usualmente una tarea paralela) en un set de nodos asignados.
3. Modera la contención de recursos gestionando una cola de trabajos en espera.

Aunque existen otros gestores de carga de trabajos, SLURM es único en varios aspectos:

**Escalabilidad:** Está diseñado para operar en un cluster heterogéneo con decenas de millones de procesadores.

**Desempeño:** Puede aceptar 1000 peticiones de trabajo por segundo y ejecutar 500 tareas simples por segundo (dependiendo del hardware y la configuración del sistema).

**Gratuito y de Código Abierto:** Su código fuente está disponible bajo la Licencia Pública General de GNU (GNU General Public License).

**Portabilidad:** Escrito en C con un motor de configuración de GNU. Si bien escrita inicialmente para Linux, SLURM ha sido portado a una amplia variedad de sistemas.

**Gestión de Energía:** Las tareas pueden especificar la frecuencia de CPU deseada y el costo energético por tarea es registrado. Recursos desocupados pueden ser apagados hasta requerirlos.

**Tolerancia a fallos:** Es muy tolerante a los fallos del sistema, incluyendo errores en la ejecución de funciones de control de nodos.

**Flexibilidad:** Existe un mecanismo de plugins para soportar diversas interconexiones, mecanismos de autenticación, planificadores, etc. Estos plugins están documentados y son bastante sencillas para el usuario final para que pueda entender las fuentes y agregar funcionalidad.

**Tareas Dimensionables:** Las tareas pueden crecer y reducirse a demanda. Las peticiones de trabajo pueden especificar rangos de tamaño y límite de tiempo.

**Trabajos de estado:** Tareas de estado en ejecución en el nivel de tareas individuales para ayudar a identificar los desequilibrios de carga y otras anomalías.

## 4.2. Comandos de SLURM

**Información General**

Existen manuales (mediante el comando `man`) para todos los demonios, comandos y funciones de APIs de SLURM. El comando `option --help`, así como `--usage` proporcionan una lista para las opciones disponibles. Nótese que todos los comandos distinguen entre mayúsculas y minúsculas. Es posible ejecutar comandos en cualquier nodo del cluster. Cualquier fallo genera un código de salida diferente de cero.

**Lista de comandos**

Los usuarios pueden interactuar con SLURM utilizando diversos comandos de terminal como por ejemplo:

**srun**: Se utiliza para enviar un trabajo para su ejecución o iniciar pasos de trabajo en tiempo real. *srun* tiene una amplia variedad de opciones para especificar los requisitos de los recursos, incluyendo: mínimo y máximo número de nodos, conteo de procesadores, y características específicas de nodos a usar (cuanta memoria, espacio en disco, ciertas características requeridas, etc.). Un trabajo puede contener varios pasos de trabajo que se ejecutan de forma secuencial o en paralelo en nodos independientes o compartidos dentro de la asignación de nodos del trabajo.

**sbcast**: Se utiliza para transferir un archivo desde un disco local a el disco local en los nodos asignados a un trabajo. Esto puede ser utilizado para ocupar eficazmente los nodos de cómputo sin disco o proporcionar un mejor rendimiento con respecto a un sistema de archivos compartidos.

**squeue**: Reporta el estado de los trabajos o pasos de trabajo. Cuenta con una amplia variedad de filtrado, clasificación y opciones de formato. Por defecto, reporta los trabajos que se están ejecutando en orden de prioridad y luego los trabajos pendientes ordenados también por prioridad.

**scancel**: Se utiliza para cancelar un paso de trabajo o un trabajo pendiente o en ejecución. También puede ser usado para enviar una señal arbitraria a todos los procesos asociados con un trabajo en ejecución o paso de trabajo.

**sinfo**: Informa el estado de las particiones y los nodos gestionados por SLURM. Cuenta con una amplia variedad de opciones de filtrado, clasificación y formato.

**sview**: Despliega la interfaz gráfica de usuario para obtener y actualizar la información del estado de los trabajos, particiones y nodos gestionados por SLURM.

**sacct**: Se utiliza para reportar la información de las cuentas de trabajos o pasos de trabajo activos o completados.

**sbatch**: Se utiliza para enviar un script de trabajo para su posterior ejecución. El script contendrá típicamente uno o más comandos *srun* para lanzar tareas paralelas.

**salloc**: Se utiliza para asignar recursos a un trabajo en tiempo real. Normalmente se utiliza para asignar recursos y generar un itérprete de comandos *shell*. Dicha terminal es utilizada entonces para ejecutar comandos *srun* para iniciar tareas en paralelo.

**sattach**: Se utiliza para vincular las entradas, salidas y errores estándar a un trabajo o paso de trabajo en ejecución. Es posible vincular y desvincularlos de trabajos múltiples veces.

**scontrol**: Es la herramienta administrativa utilizada para ver y/o modificar el estado SLURM. Tenga en cuenta que muchos comandos *scontrol* sólo se pueden ejecutar como usuario *root*.

**smap**: Reporta la información del estado de los trabajos, particiones y nodos gestionados por SLURM, pero muestra gráficamente la información para reflejar la topología de red.

**strigger**: Se utiliza para establecer, obtener o ver activadores de eventos. Los disparadores de eventos incluyen cosas tales como bajar nodos o trabajos que se acercan a su límite de tiempo.

**sacctmgr**: Se utiliza para ver y modificar la información de las cuentas en SLURM. Se utiliza con el demonio *slurmdbd.*

**Información básica para la carga de trabajos en SLURM**

Los siguientes son los comandos e información básica para que usuarios del LNS puedan ejecutar programas en SLURM:

- Se requiere crear un archivo denominado *job script* (sin una extensión de fichero específica) con los detalles de su cálculo y enviarlo al sistema de colas SLURM mediante la orden:

  ```
  sbatch job_script
  ```

  donde job_script es el nombre de su archivo creado.

- Para monitorear sus tareas (*jobs*) en SLURM se realiza con el comando

  ```
  squeue -u user_name
  ```

  donde `user_name` es el nombre de usuario con el que está registrado en el sistema.

- Para cancelar alguna tarea se hace con el comando

  ```
  scancel job_id
  ```

  donde `job_id` es el identificador que aparece en la salida de `squeue`

## 4.3. Scripts para ejecución de programas en SLURM

Un script es un pequeño archivo que contiene comandos para que un intérprete pueda ejecutarlos. En el caso de los scripts para SLURM, incluye tanto comandos para terminal que ya conoce como directivas para que SLURM configure la forma en que sus cálculos se ejecutarán en el cluster de supercómputo. Para hacer uno, simplemente abra su editor de texto preferido, agregue los comandos e instrucciones necesarias y guarde el archivo sin una extensión específica. Para ejecutarlo, utilice el comando `sbatch`.

Es sumamente importante que mande a ejecutar sus cálculos a través de un *script* de SLURM, ya que sólo de ésta manera podrá hacer uso del poder de procesamiento de la supercomputadora del LNS.

En ésta sección podrá encontrar un conjunto de scripts para ejecutar algunos de los programas LAMMPS, Quantum Espresso, SIESTA, GROMACS y Gaussian, disponibles en el LNS, los cuales ya han sido probados por usuarios reales y utilizados para la carga de tareas en la cola de trabajos (*queue*) de SLURM.

### 4.3.1. Script genérico

Si entre las sugerencias disponibles no encuentra el script para un programa específico, puede utilizar la siguiente estructura genérica:

```
#!/bin/bash
#SBATCH -J test          # job name
#SBATCH -o test.o%j      # output and error file name (%j expands to
jobID)
#SBATCH -n 1             # number of MPI tasks (cores) requested
#SBATCH --ntasks-per-node=1   # task (cores) per node (maximum 24)
#SBATCH -p comp               # SLURM queue (partition)
#SBATCH -t 01:00:00           # run time (hh:mm:ss)
echo $SLURM_JOB_ID
echo $SLURM_JOB_NAME
echo $SLURM_JOB_NUM_NODES
# Load your modules here
module load compilers/intel/parallel_studio_xe_2015/15.0.1
module load tools/intel/impi/5.0.2.044

# Run your task here
mpirun -genv I_MPI_FABRICS shm:ofa YOUR_TASK
```

donde solo queda por especificar la cantidad de nodos y *cores* que utilizará para su cálculo, los módulos de software que utiliza y la ejecución final de su código en la última línea.

### 4.3.2. Uso de LAMMPS

El siguiente ejemplo es un *job script* para correr LAMMPS en 48 cores (2 nodos) del cluster de supercómputo.

```
#!/bin/bash
#SBATCH -J lmp      # job name
#SBATCH -o lmp.o%j  # output and error file name (%j expands to
jobID)
#SBATCH -n 48       # total number of MPI tasks (cores) requested
#SBATCH --ntasks-per-node=24  # task (cores) per node (maximum 24)
#SBATCH -p comp               # SLURM queue (partition)
#SBATCH -t 24:00:00           # run time (hh:mm:ss)

# Load Intel Parallel Studio
module load compilers/intel/parallel_studio_xe_2015/15.0.1
module load tools/intel/mkl/11.2.1
module load tools/intel/impi/5.0.2.044
module load applications/lammps/10_aug_2015

# Run the LAMMPS task
mpirun -genv I_MPI_FABRICS shm:ofa $LMP < input > output
```

A partir de éste *script*, solo queda sustituir en la última línea los nombres de los archivos de entrada (`input`) y salida (`output`) de su cálculo, y de ser necesario, el número de *cores* y tiempo de ejecución requeridos.

### 4.3.3. Uso de Quantum Espresso

A continuación se presenta un ejemplo de *job script* para correr Quantum Espresso usando 4 nodos de cálculo con 24 cores cada uno (96 cores en total):

```
#!/bin/bash
#SBATCH -J pw         # job name
#SBATCH -o pw.o%j     # output and error file name (%j expands to
jobID)
#SBATCH -n 96  # total number of MPI tasks requested (maximum 240)
#SBATCH --ntasks-per-node=24  # number of tasks per node (maximum
24)
#SBATCH -p comp               # SLURM queue (partition)
#SBATCH -t 24:00:00           # run time (hh:mm:ss)

# Load Intel Parallel Studio
module load compilers/intel/parallel_studio_xe_2015/15.0.1
module load tools/intel/mkl/11.2.1
module load tools/intel/impi/5.0.2.044
```

```
module load applications/quantum-espresso/5.1.2

# run the Quantum Espresso executable
mpirun -np 96 -genv I_MPI_FABRICS shm:ofa $PW -i input > output
```

A partir de éste *script*, solo queda sustituir en la última línea los nombres de los archivos de entrada (`input`) y salida (`output`) de su cálculo, y de ser necesario, el número de *cores* y tiempo de ejecución requeridos.

### 4.3.4. Uso de SIESTA

El siguiente ejemplo es un *job script* para correr SIESTA en 24 cores (1 nodo) del cluster de supercómputo.

```
#!/bin/bash
#SBATCH -J siesta      # job name
#SBATCH -o siesta.o%j  # output and error file name (%j expands to
jobID)
#SBATCH -n 24          # total number of MPI tasks requested
#SBATCH --ntasks-per-node=24  # number of tasks per node (maximum
24)
#SBATCH -p comp               # SLURM queue (partition)
#SBATCH -t 02:00:00           # run time (hh:mm:ss)

# Load Intel Parallel Studio
module load compilers/intel/parallel_studio_xe_2015/15.0.1
module load tools/intel/mkl/11.2.1
module load tools/intel/impi/5.0.2.044

# Run the parallel siesta executable
mpirun -np 24 -genv I_MPI_FABRICS shm:ofa /software/LNS/siesta
/3.2/bin/siesta < input.fdf
```

A partir de éste *script*, solo queda sustituir en la última línea los nombres de los archivos de entrada (`input`) y salida (`output`) de su cálculo, y de ser necesario, el número de *cores* y tiempo de ejecución requeridos.

### 4.3.5. Uso de GROMACS

A continuación se presenta un ejemplo de *job script* para correr GROMACS usando 4 nodos de cálculo con 24 cores cada uno (96 cores en total):

```
#!/bin/bash
#SBATCH -J gromacs     # job name
```

```
#SBATCH -o gromacs.o%j # output and error file name (%j expands to
jobID)
#SBATCH -n 96          # total number of mpi tasks requested
#SBATCH --ntasks-per-node=24  # number of tasks per node
#SBATCH -p comp               # SLURM queue (partition)
#SBATCH -t 120:00:00          # run time (hh:mm:ss)
module load compilers/intel/parallel_studio_xe_2015/15.0.1
module load tools/intel/mkl/11.2.1
module load tools/intel/impi/5.0.2.044
source /software/LNS/gromacs/5.0.4/bin/GMXRC.bash

# run the GROMACS task
mpirun -genv I_MPI_FABRICS shm:ofa -np 96 mdrun_mpi -deffnm
min.tpr
```

A partir de éste *script*, solo queda sustituir en la última línea los nombres de los archivos de entrada (`input`) y salida (`output`) de su cálculo, y de ser necesario, el número de *cores* y tiempo de ejecución requeridos.

### 4.3.6. Uso de Gaussian

Para utilizar Gaussian de manera optima es necesario considerar el hardware donde se va a correr. Esto es para determinar qué modo de ejecución (memoria compartida o memoria distribuida o ambos) es más conveniente para el problema que se está tratando y para estimar los requerimientos de memoria, almacenamiento y tiempo de cálculo. Por lo tanto es importante conocer el escalamiento del método de estructura electrónica que se utiliza en función del número total de funciones base.

Es bien sabido, por ejemplo, que DFT tiene un escalamiento de memoria del orden de 3N^2 bytes, donde N es el número de funciones base. Asimismo, DFT tiene un escalamiento de espacio de disco (para almacenar las integrales) del orden de N^2 bytes. Tradicionalmente, la compañía Gaussian Inc. no ofrece información sobre el funcionamiento interno del código Gaussian para determinar con más certeza estos requerimientos, así que es necesario usar la experiencia para estimarlos.

En el caso especifico del hardware del LNS, cada nodo de cálculo posee 24 cores de ejecución y 128 GB de memoria (que para fines prácticos se puede usar al 85%, o sea, del orden de 110 GB). Es importante no sobrepasar este monto de memoria. Por lo tanto, hay que declarar siempre la memoria a utilizar en el archivo de entrada de Gaussian (opcion %mem) teniendo en cuenta la regla $3N^2$.

A continuación se presenta un ejemplo de *job script* para correr Gaussian para procesar una molécula del orden de 60 átomos usando el modo de ejecución mixto (memoria distribuida + memoria compartida) utilizando 5 nodos de cálculo con 12 cores cada uno (60 cores en total) usando mediante el sistema de comunicación TCP Linda:

```
#!/bin/bash
#SBATCH -J gaussian     # job name
#SBATCH -o gaussian.o%j # output and error file name (%j expands
to jobID)
#SBATCH -n 60           # total number of multicore tasks
requested
#SBATCH --ntasks-per-node=12    # number of tasks per node
(maximum 24)
#SBATCH -p comp                 # SLURM queue (partition)
#SBATCH -t 72:00:00             # run time (hh:mm:ss)

# Initialize Gaussian environment variables

module load applications/gaussian/09
ulimit -c 0
ulimit -d hard
ulimit -f hard
ulimit -l hard
ulimit -m hard
ulimit -n hard
ulimit -s hard
ulimit -t hard
ulimit -u hard
export GAUSS_SCRDIR=/home/$USER/tmp/$SLURM_JOBID
mkdir -p $GAUSS_SCRDIR

# Specify input and output files

INPUT=input.com
OUTPUT=output.out

# Append the list of Linda workers at the beginning of input file

nodelist=`scontrol show hostname $SLURM_NODE_LIST | tr '\n' ','`
nodelist=${nodelist%,}
sed "/LindaWorkers/d" $INPUT > ${INPUT}.tmp
sed -i "1i %UseSSH" ${INPUT}.tmp
sed -i "1i %LindaWorkers=$nodelist" ${INPUT}.tmp
sed -i "1i %NProcShared=12" ${INPUT}.tmp

# Run Gaussian
$g09root/g09/g09 < ${INPUT}.tmp > ${OUTPUT}
```

```
# Clean out garbage
rm -rf ${INPUT}.tmp
```

A partir de éste *script*, solo queda especificar los archivos de entrada y salida en las líneas `INPUT=input.com` y `OUTPUT=output.out` con los nombres de archivo deseados para cálculo, y de ser necesario, el número de *cores* y tiempo de ejecución solicitados.

Además, para este ejemplo, es importante que no incluya línea que contenga la orden `%NProcShared` o `%LindaWorkers` de su fichero de entrada de Gaussian; y que en la línea de especificación de memoria coloque un valor apropiado para su cálculo, digamos para éste ejemplo, de `%mem=32gb`

## 5. Información adicional

En caso de requerir más información relativa a los tópicos descritos en ésta guía, es posible encontrar gran cantidad de recursos en la red y/o en otras guías proporcionadas por el LNS. Si desea obtener ayuda personalizada con respecto al manejo de su cuenta o la ejecución de sus cálculos, también puede dirigirse a los correos de atención al usuario proporcionados en el sitio web oficial del LNS (www.lns.buap.mx).