ÁREA DE TECNOLOGÍA Y COMUNICACIONES

SSC for Newbies II

Servicio de Supercomputación
Unidad de Sistemas y Redes

Octubre 2016

Control de versiones

DOCUMENTO/ARCHIVO

Título:	Manual SSC para principiantes
Código:	TIC-MAN-SSC_II
Fecha:	octubre de 2016
Versión:	1.0
Ubicación física:	

REGISTRO DE CAMBIOS

Versión	Fecha	Motivo del cambio
0.1	20.09.20016	Primera revisión
		Autor: Rafael Espinosa González
0.2	28/09/2016	Revisores: Susana Leal Escobar y Francisco Sanchez Alguacil

Indice

1 Introducción

El presente documento pretende ser una guía rápida para la correcta utilización de los recursos que la Universidad de Castilla-La Mancha pone a disposición de sus investigadores a través del Servicio de Supercomputación del Área TIC.
A lo largo del mismo se ofrece la información necesaria para poder acceder correctamente a los recursos así como la utilización de los mismos en la ejecución de trabajos.

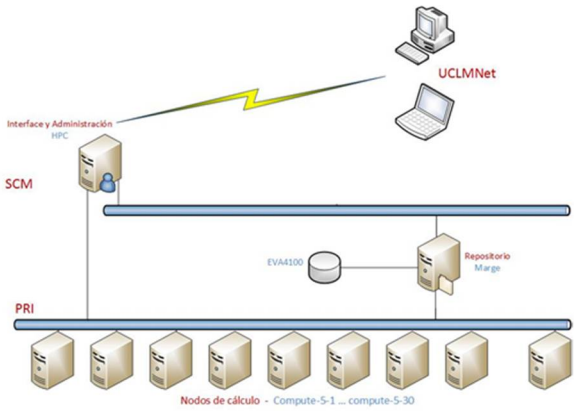
2 Recursos

2.1 Clústeres

En el servicio de SuperComputación de la Universidad de Catilla-La Mancha (en adelante SSC) en el momento de escribir el presente manual están operativos tres sistemas de cómputo o tres clústeres.

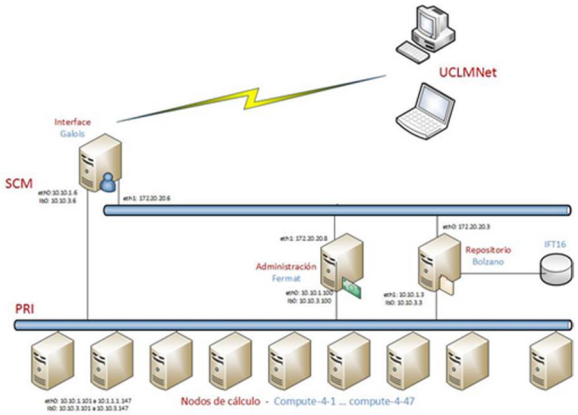
Todo ellos mantienen semejante arquitectura, aunque con particularidades. Pasamos a conocerlos:

Clúster HPC



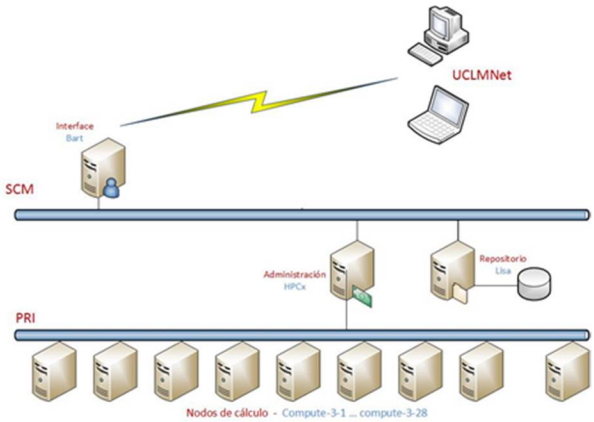
- 30 Nodos de cálculo con 2 Xeon E5472Q
- 1 Nodo de Acceso + 1 Nodo de servicio de disco con 2 Xeon E5472Q
- Cada nodo de cálculo con
  - 2 x GEthernet
  - Infiniband 20 gbps
  - 16 GB DDR2
  - HD 160GB SATA
- FS Compartido 10TB

Clúster Thales



- 34 Nodos de cálculo con 2 Xeon E5472Q
- 11 Nodos de cálculo con 2 Xeon E5450Q
- 1 Nodo de Acceso + 1 Nodo de Control con 2 Xeon E5472Q
- 1 Nodo servidor de disco con 2 Xeon E5645
- Cada nodo de cálculo con
  - 2x GEthernet
  - Infiniband 20 gbps
  - 16 GB DDR2
  - HD 160GB SATA
- FS Compartido 5TB

Clúster HPCx



- 28 Nodos de cálculo con 2 Xeon E5645
- 1 Nodo de Control con 2 Xeon E5645
- 1 Nodo de Acceso con 2 Xeon E5630
- 1 Nodo de servicio de disco con 2 Xeon E5472
- Cada nodo de cálculo con
  - 2x GEthernet
  - Infiniband 20 gbps
  - 48 GB DDR2
  - HD 500GB SATA
- FS Compartido 4TB

2.2 Roles

Ya hemos comentados que los tres sistemas que ofrecen recursos en SSC tienen la misma arquitectura. En dicha arquitectura podemos identificar los siguientes roles:

Servicio de Acceso – Equipo que dedica sus recursos a recibir conexiones desde el exterior. Dicho de manera menos técnica, es el equipo que recibe nuestras conexiones ya sean por ssh o por sftp.

Servicio de Disco – Equipo que proporciona disco al resto de equipos del clúster. Este servicio recibe las conexiones de cada nodo del clúster y les proporciona almacenamiento. El hecho de ser el único punto de almacenamiento del clúster hace que la visión del almacenamiento se mantenga coherente en todo el clúster.

Control del clúster – Equipo que se dedica a controlar el clúster. Administra colas, asigna tareas para ejecución, localiza nodos con recursos disponibles y algunas otras tareas

Nodos de Cálculo – Son, por último, los equipos que se dedican al cálculo en sentido estricto.

2.3 Equipos

Ya hemos visto en apartados anteriores los distintos clústeres y los distintos roles. Hay que remarcar que un equipo físico puede desempeñar uno o varios roles dependiendo de la disponibilidad de equipos en el momento de construir el clúster. Vemos en la siguiente tabla de recursos del SSC qué equipos corresponden a qué clúster y con qué rol concreto.

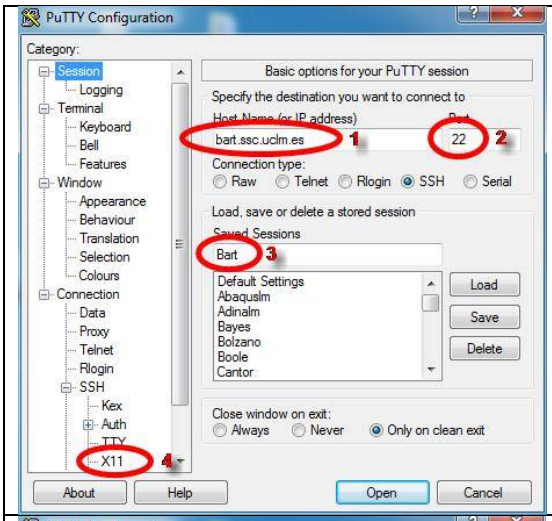
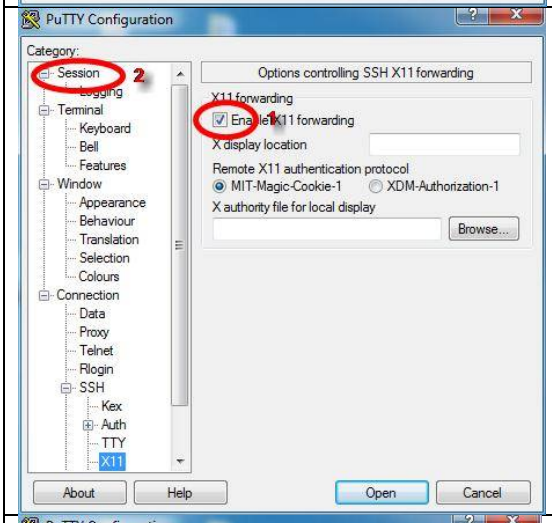
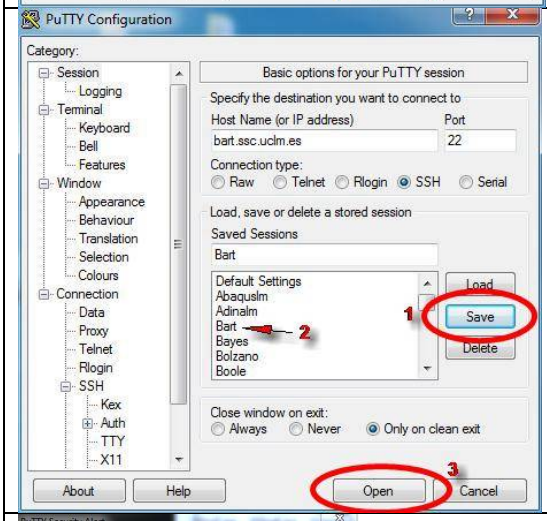
	Cluster HPC	Cluster HPCx	Cluster Thales
S. Acceso	hpc.uclm.es	bart.ssc.uclm.es	galois.ssc.uclm.es
S. Disco	marge.ssc.uclm.es	lisa.ssc.uclm.es	bolzano.ssc.uclm.es
S. Control	hpc.uclm.es	hpcx.ssc.uclm.es	fermat.ssc.uclm.es
N. Cálculo	compute-5-1 a compute-5-30	compute-3-1 a compute-3-28	compute-4-1 a compute-4-34

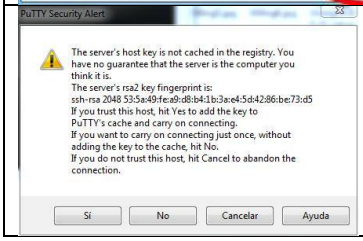
3 Configuración y Acceso

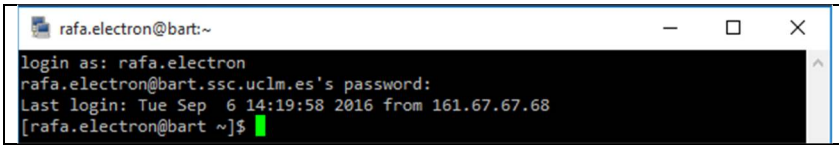
3.1 Escritorio Windows

3.1.1 SSH

Para acceder desde un entorno Windows existen diversas alternativas en la red tanto en software privativo como de fuentes abiertas. Nosotros vamos a utilizar el conocido programa Putty que se distribuye bajo licencia Open Source. Una vez instalado, iniciamos el programa y procedemos a configurar una nueva conexión según la secuencia siguiente:

<table>
<tr>
<td> 
</td>
<td>1- Nombre del servidor de acceso a conectar (bart.ssc.uclm.es)
2- Puerto de conexión (22)
3- Nombre que le damos a la conexión (Bart)
4- Nos vamos al menú X11 para continuar con la definición de una conexión</td>
</tr>
<tr>
<td> 
</td>
<td>1- Activar el reenvío del tráfico generado por el protocolo X11
2- Volvemos al apartado de Sesión</td>
</tr>
<tr>
<td> 
</td>
<td>1- Salvamos la conexión configura
2- Nos aparece un nuevo ítem en la lista de conexiones con el nombre que le pusimos en el paso 3 de la primera imagen
3- Tras marcar esa nueva conexión, basta ahora abrir la conexión para enlazar con el servidor de acceso pretendido.

• Existen muchas opciones en la configuración de una sesión pero nos limitamos a comentar las que nos resultan imprescindibles.</td>
</tr>
<tr>
<td> 
</td>
<td>1- Esta ventana aparece la primera vez que accedemos al pretendido servidor. Nos presenta la huella digital del mismo y nos pide confirmación para guardarla.</td>
</tr>
</table>

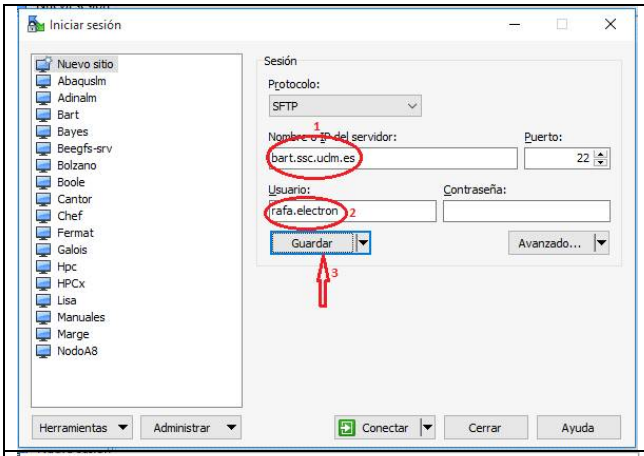
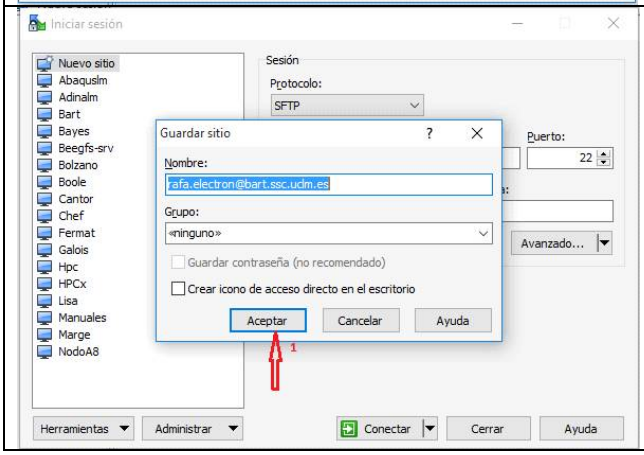
	Nos pide usuario y contraseña y hacemos login.

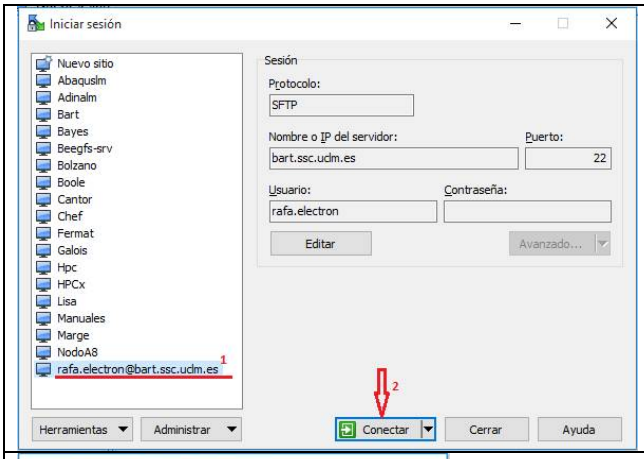
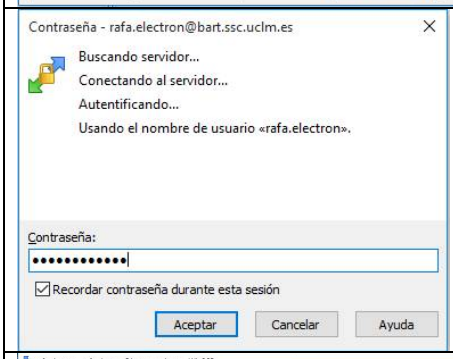
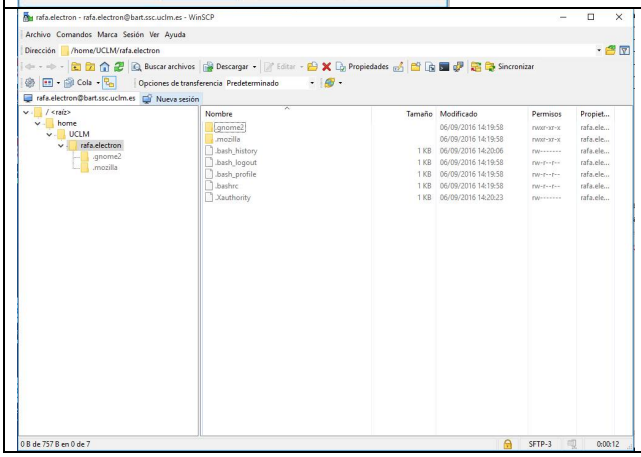
Intencionadamente hemos dejado sin comentar el apartado 1 de la segunda imagen. Lo comentaremos en el apartado 3.1.3 aunque debemos dejar expresamente repetido que son configuraciones imprescindibles.

3.1.2 SFTP

Vamos, a continuación, a configurar una conexión para el uso el protocolo SFTP que nos permita intercambiar ficheros entre nuestro escritorio y el servidor de acceso. En algún apartado anterior ya comentamos que el espacio de repositorio es único y que todos los equipos de cada clúster se conectan a ese repositorio por lo que el acceso que nos ocupa nos servirá tanto para dejar en el clúster fichero con parámetros de ejecución como para recoger desde el servidor ficheros de resultados.

De entre los diversos programas capaces de conectar con el citado protocolo, elegimos WinSCP. Una vez instalado, lo ejecutamos y pasamos a definir una sesión.

	1- Nombre del servidor de acceso a conectar (bart.ssc.uclm.es) 2- Usuario (rafa.electron) 3- Hacemos click sobre el botón Guardar
	1- Nos propone un nombre de la sesión generado a partir del equipo destino y el nombre de usuario y hacemos click sobre el botón Aceptar (rafa.electron@bart.ssc.uclm.es)

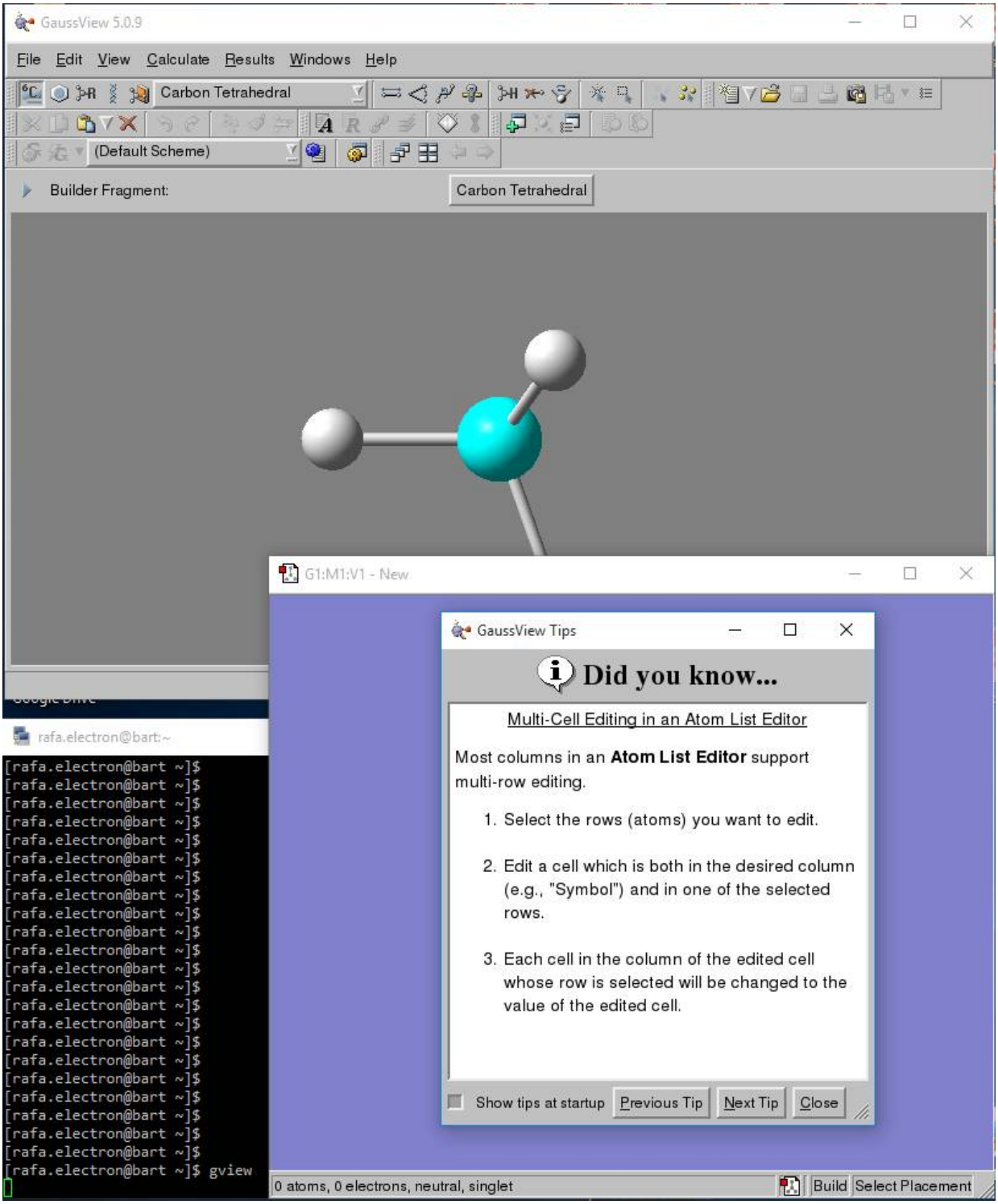
	1- En la parte izquierda de la ventana nos ha añadido una nueva sesión (rafa.electron@bart.ssc.uclm.es) 2- Hacemos click sobre el botón Conectar
	1- Nos pide la contraseña
	1- Aparece la ventana de trabajo. Para enviar fichero al servidor de acceso basta arrastrarlos hasta esta ventana. Por el contrario, para recuperar ficheros desde el servidor de acceso basta arrastrarlos desde el mismo a nuestro escritorio

3.1.3 X Windows

Trabajar con entorno X Windows es más sencillo puesto que lo único que tenemos que hacer es arrancar el servicio en nuestro escritorio para que quede a la escucha de las peticiones de ventana que pueda recibir. En este momento podemos comentar algo que habíamos dejado pendiente en el apartado 3.1.1 en el apartado 1 de la 2ª imagen. El entorno gráfico remoto sólo funcionará si en la definición de la sesión que utilizamos para conectar y ejecutar el programa que necesita entorno gráfico hemos activado la casilla “Enable X11 Forwarding”. Esta opción es propia de la especificación del protocolo desde sus primeras revisiones por lo que debe aparecer en cualquier software que soporte conexiones ssh.

Basta pues con arrancar el programa XMing, asegurar que nos aparece el icono correspondiente en la parte inferior derecha de nuestro escritorio, arrancar una sesión y ejecutar algún programa que requiera entorno gráfico.

	1- Icono de XMing



Hemos ejecutado el programa gview que genera una serie de ventanas (3) en nuestro escritorio.

3.2 Escritorio Linux

3.2.1 SSH

Para poder ejecutar el cliente ssh es necesario tener instalado el paquete **openssh-client** (Ubuntu) o el paquete **openssh-clients** (CentOS) con sus dependencias en cada caso.

Para establecer una conexión basta ejecutar el comando ssh con el siguiente formato:

$ ssh –X [usuario@]equipo

En nuestro caso el nombre de usuario corresponde a un usuario de la Intranet de la UCLM. El parámetro -X cumple la misma función que la casilla que veíamos en la imagen 2, punto 1 del apartado 3.1.1 anterior (Enables X11 Forwarding).

```
root@manuales:/root# ssh -X rafa.electron@bart.ssc.uclm.es
The authenticity of host 'bart.ssc.uclm.es (172.20.50.11)' can't be established.
RSA key fingerprint is 53:5a:49:fe:a9:d8:b4:1b:3a:e4:5d:42:86:be:73:d5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'bart.ssc.uclm.es,172.20.50.11' (RSA) to the list of known ho
sts.
rafa.electron@bart.ssc.uclm.es's password:
Last login: Wed Sep  7 14:25:29 2016 from 161.67.67.68
[rafa.electron@bart ~]$
```

En el ejemplo vemos que, por ser la primera vez que se realizar esa conexión, antes de solicitar la contraseña pide la aceptación de la huella digital del equipo destino.

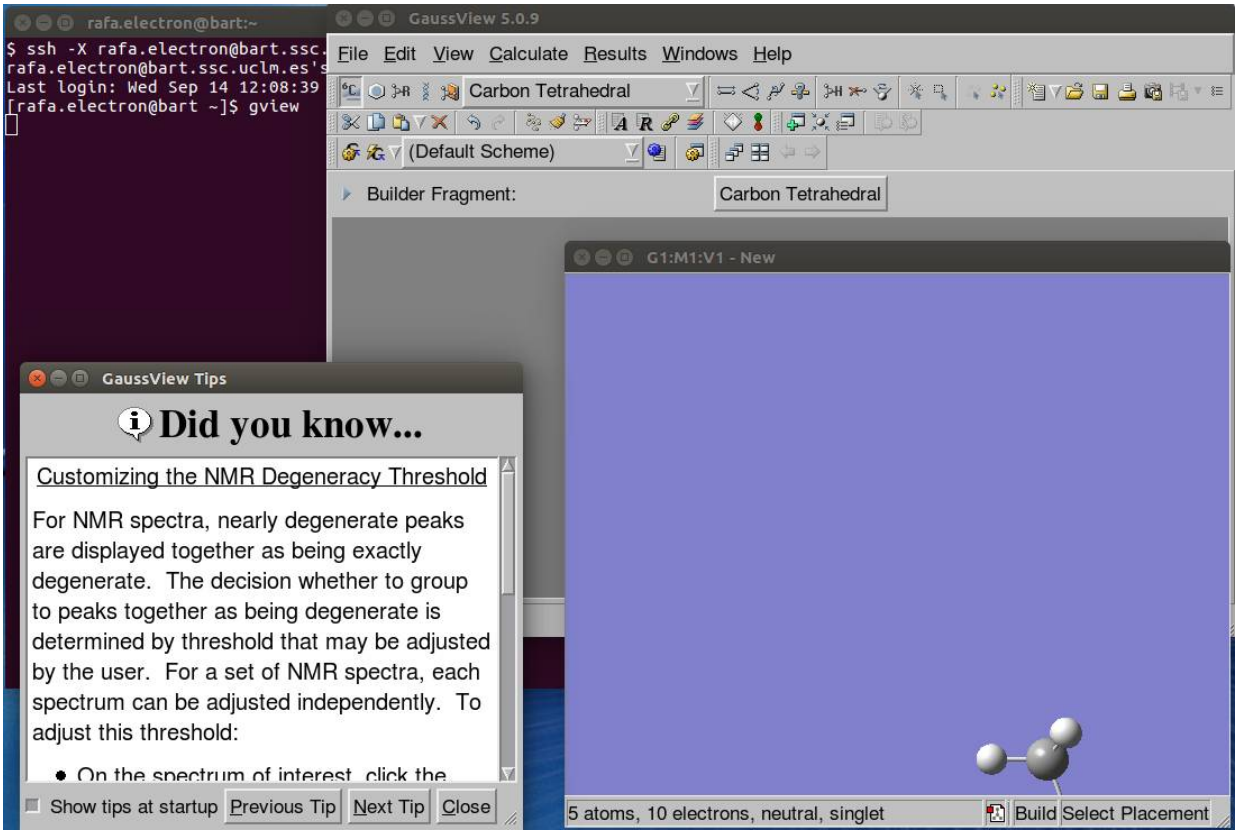
3.2.2 SFTP

El cliente SFTP estará disponible a partir de los paquetes instalados en el apartado anterior. Basta ejecutar el comando correspondiente para acceder a un intérprete de comandos propio dedicado a la trasferencia de ficheros.

```
root@manuales:/root# sftp  rafa.electron@bart.ssc.uclm.es
Connecting to bart.ssc.uclm.es...
rafa.electron@bart.ssc.uclm.es's password:
sftp>
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
get [-P] remote-path [local-path]  Download file
help                               Display this help text
lcd path                           Change local directory to 'path'
lls [ls-options [path]]            Display local directory listing
lmkdir path                        Create local directory
ln oldpath newpath                 Symlink remote file
lpwd                               Print local working directory
ls [-1aflnrSt] [path]              Display remote directory listing
lumask umask                       Set local umask to 'umask'
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-P] local-path [remote-path]  Upload file
pwd                                Display remote working directory
quit                               Quit sftp
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell
?                                  Synonym for help
```

3.2.3 XWindows

Para establecer una sesión X Windows seguimos un procedimiento semejante al que utilizamos en el apartado 3.1.3 para entorno Windows. Necesitamos establecer una sesión SSH en la que hayamos activado el reenvío X11. Una vez establecida la sesión basta lanzar el programa con entorno gráfico y aparecerá en nuestro escritorio local la o las ventanas pertinentes. En nuestro caso no hemos tenido que instalar paquete alguno para que el entorno gráfico funcione aunque esto puede variar dependiendo de la distribución instalada y de la configuración de nuestro sistema.



3.3 Escritorio MAC

3.3.1 SSH

El cliente ssh está integrado en la aplicación **Terminal** que lo podemos encontrar siguiendo la secuencia **Finder – Aplicaciones – Utilidades – Terminal**

El intercambio es completamente semejante al correspondiente al apartado anterior 3.2.1

```
UGIC-AB-2-MAC:~ ugicc$ ssh -X rafa.electron@bart.ssc.uclm.es
The authenticity of host 'bart.ssc.uclm.es (172.20.50.11)' can't be established.
RSA key fingerprint is 53:5a:49:fe:a9:d8:b4:1b:3a:e4:5d:42:86:be:73:d5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'bart.ssc.uclm.es,172.20.50.11' (RSA) to the list of
known hosts.
rafa.electron@bart.ssc.uclm.es's password:
Last login: Wed Sep 14 12:36:40 2016 from rafael-espin-pt.uclm.es
[rafa.electron@bart ~]$
```

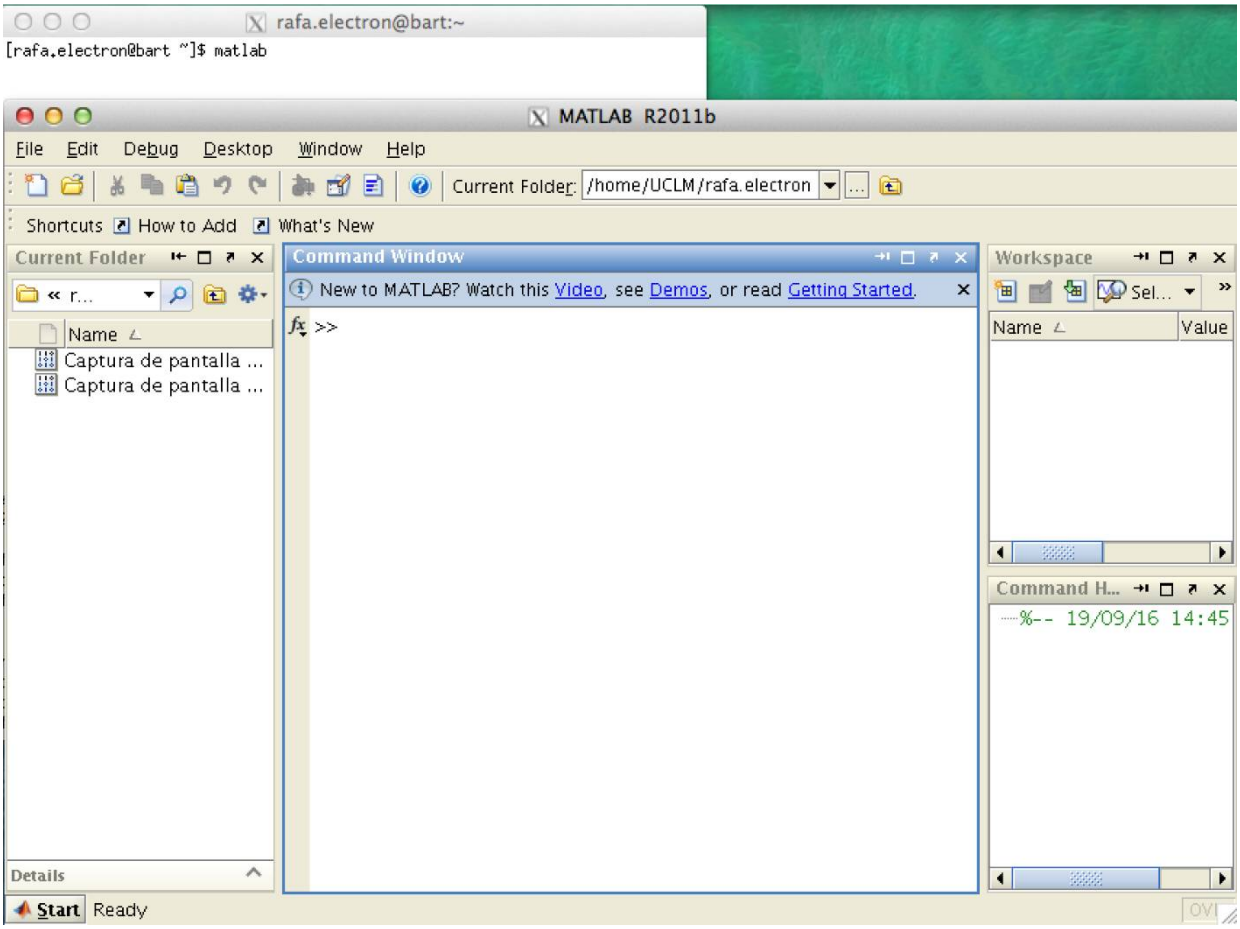
3.3.2 SFTP

Al igual que el cliente SSH, el cliente SFTP también está integrado en la aplicación Terminal

```
UGIC-AB-2-MAC:~ ugicc$ sftp rafa.electron@bart.ssc.uclm.es
rafa.electron@bart.ssc.uclm.es's password:
Connected to bart.ssc.uclm.es.
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
get [-Ppr] remote [local]          Download file
help                               Display this help text
```

3.3.3 XWindows

Antes de establecer una sesión XWindows es necesario instalar en nuestro sistema IOS un servicio X11. Este servicio no forma parte del sistema, pero el propio IOS nos indica cómo conseguirlo. Hay que ir a Finder –> Utilidades -> X11 para que nos indique donde conseguir un servidor X11. Posteriormente hay que instalar XQuartz bajado desde www.xquartz.org e iniciar el programa. Para establecer una sesión X Windows seguimos un procedimiento semejante al que utilizamos en los apartados anteriores 3.1.3 y 3.2.3. Necesitamos establecer una sesión SSH en la que hayamos activado el reenvío X11. Esto lo conseguimos utilizando en el comando ssh con el parámetro -X. Una vez establecida la sesión basta lanzar el programa con entorno gráfico y aparecerá en nuestro escritorio local la o las ventanas pertinentes.



4 Tareas y trabajos

A lo largo de esta sección se instruirá en el manejo del software OpenLava, el gestor de colas instalado en los clústeres operativos en el SSC.

4.1 El gestor de trabajos LSF

OpenLava es el software que gestiona y optimiza dónde se ejecutarán los proyectos en base a los recursos informáticos de SSC, permitiendo un eficiente uso de los mismos mediante técnicas de balanceo de carga. Se adapta perfectamente a la arquitectura de memoria compartida de los nodos de cómputo compute-n-m y es capaz de sacarles el máximo partido.

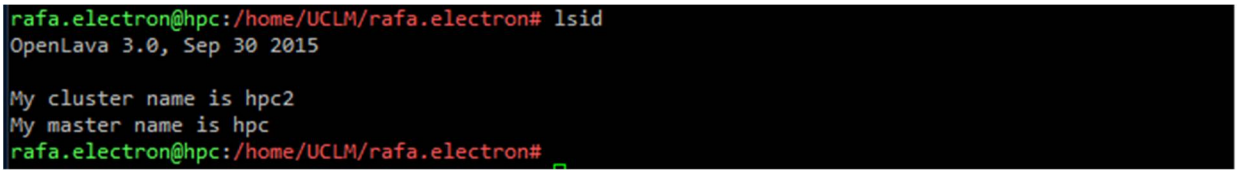
También permite conocer en todo momento el estado del clúster y elige de forma adecuada dónde se ejecutarán los siguientes trabajos.

Mediante la utilización de OpenLava, y a través de los siguientes apartados, se descubrirá el método de trabajo del gestor de colas y su organización, de forma que se aprenderá a lanzar trabajos y a obtener diversa información sobre el estado de los trabajos.

4.2 Estado del cluster

Antes de comenzar a trabajar con uno de los clústeres del SSC, puede interesarnos conocer el estado del mismo. Hay algunos comandos especialmente interesantes: lsid, lshosts y bhosts.

El comando **"lsid"** nos permite identificar el nombre del clúster según el software OpenLava, el nombre del master del clúster y la versión de software utilizada.



El comando **"lshosts"** nos presenta los componentes del clúster según la definición y detección de OpenLava.

```
root@hpcx:/root# lshosts -w
HOST_NAME                      type       model  cpuf ncpus maxmem maxswp server RESOURCES
hpcx                        LINUX64        BULL 100.0    12 48258M 51146M    Yes (cs)
compute-3-1                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-2                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-3                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-4                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-5                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-6                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-7                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-8                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-9                 LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-10                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-11                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-12                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-13                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-14                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-15                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-16                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-17                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-18                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-19                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-20                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-21                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-22                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-23                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-24                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-25                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-26                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-27                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
compute-3-28                LINUX64        BULL 100.0    12 48390M 47999M    Yes (cs)
root@hpcx:/root#
```

Por último, el comando bhosts nos muestra el nivel de ocupación actual de los componentes del clúster

```
root@hpcx:/root# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
compute-3-1        closed          -     12     12     12      0      0      0
compute-3-10       ok              -     12      0      0      0      0      0
compute-3-11       closed          -     12      0      0      0      0      0
compute-3-12       ok              -     12      6      6      0      0      0
compute-3-13       ok              -     12      0      0      0      0      0
compute-3-14       ok              -     12      0      0      0      0      0
compute-3-15       ok              -     12      0      0      0      0      0
compute-3-16       ok              -     12      2      2      0      0      0
compute-3-17       ok              -     12      0      0      0      0      0
compute-3-18       ok              -     12      0      0      0      0      0
compute-3-19       ok              -     12      0      0      0      0      0
compute-3-2        closed          -     12     12     12      0      0      0
compute-3-20       ok              -     12      6      6      0      0      0
compute-3-21       ok              -     12     11     11      0      0      0
compute-3-22       ok              -     12      0      0      0      0      0
compute-3-23       ok              -     12      0      0      0      0      0
compute-3-24       closed          -     12     12     12      0      0      0
compute-3-25       ok              -     12      0      0      0      0      0
compute-3-26       ok              -     12      0      0      0      0      0
compute-3-27       closed          -     12     12     12      0      0      0
compute-3-28       ok              -     12      0      0      0      0      0
compute-3-3        ok              -     12      0      0      0      0      0
compute-3-4        closed          -     12     12     12      0      0      0
compute-3-5        closed          -     12     12     12      0      0      0
compute-3-6        ok              -     12      6      6      0      0      0
compute-3-7        ok              -     12      6      6      0      0      0
compute-3-8        ok              -     12      6      6      0      0      0
compute-3-9        ok              -     12      0      0      0      0      0
hpcx               closed          -      6      0      0      0      0      0
root@hpcx:/root#
```

4.3 Las colas de trabajos

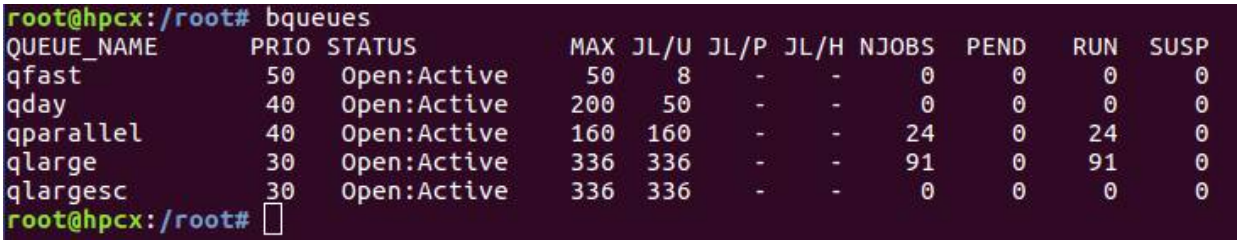
El software OpenLava trabaja según el paradigma de colas. De esta forma los trabajos se lanzan a una cola que se encarga de localizar los recursos libres solicitados para definitivamente asignar la ejecución de la tarea. Siempre hay una cola que recibe la petición de tarea. Con ese fin se utiliza el concepto de cola por defecto. Cada clúster tiene una cola por defecto. Si en el comando no explicitamos la cola a la que enviamos la tarea, el sistema asigna la tarea a la cola por defecto.

Esta aproximación permite un máximo aprovechamiento de los recursos de cómputo disponibles, ya que cada máquina del clúster tendrá capacidad de ejecución de trabajos de acuerdo a sus características físicas (número de procesadores, memoria total, espacio de swap y en disco...), así como permitirá el máximo aprovechamiento de los sistemas en términos de tiempo, ya que al finalizar una tarea podrá lanzar la siguiente en el equipo cuyos recursos hayan sido liberados.

Cada cola tiene asignados unos equipos de ejecución mientras que cada nodo de ejecución deberá estar asignado al menos a una cola. De no ser así, ese equipo no recibirá tareas a ejecutar. En nuestro caso, como todos los equipos tienen la misma arquitectura, todos los equipos están asignados a todas las colas. En el caso del clúster HPCx que maneja varias colas, éstas se diferencian por la prioridad y los límites de tiempo en ejecución.

La definición de colas para cada clúster es dinámica. Esto es, se puede modificar en cualquier momento y según necesidad. En el momento de escribir este documento, las colas definidas en cada clúster se ven en las siguientes imágenes:

Clúster HPCx



Clúster Thales

```
root@fermat:/root# bqueues -l normal

QUEUE: normal
  -- For normal low priority jobs, running only if hosts are lightly loaded.  Th
is is the default queue.

PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 40   20  Open:Active       -    -    -    -    52     0    52     0     0    0
Interval for a host to accept two jobs is 0 seconds

SCHEDULING PARAMETERS
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

USERS:  all users
HOSTS:  nodos/
PRE_POST_EXEC_USER:  root
PRE_EXEC:  ulimit -l unlimited
POST_EXEC:  rm -rf /scratch/$LSB_JOBID
RES_REQ:  span[hosts=1]
JOB_STARTER:  /bin/bash -c "mkdir /scratch/$LSB_JOBID;GAUSS_SCRDIR=/scratch/$LSB
_JOBID; %USRCMD "


root@fermat:/root#
```

Clúster HPC

```
root@hpc:/root# bqueues
QUEUE_NAME      PRIO      STATUS      MAX  JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
normal           40    Open:Active      -  200    -    -   134     0   134     0
root@hpc:/root#
```

4.4 Estado de un trabajo

Para monitorizar el estado de una tarea utilizaremos principalmente el comando bjobs con sus principales parámetros

$ bjobs [-u <user> | -u all] -l –w [<jobid>]

Parámetro	**Significado**
-u <user>	Tareas en estado de ejecución del usuario <user>
-u all	Tareas en estado de ejecución de todos los usuarios
-w	Salida extendida sin acortar los valores de los campos
-l	Salida en formato largo con toda la información de la tarea
<jobid>	Identificador de la tarea que queremos monitorizar

Veamos unos ejemplos sencillos.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub uname;bjobs
Job <7215> is submitted to default queue <normal>.
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7215    rafa.el PEND  normal     hpc                     uname      Sep  1 11:23
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

En el ejemplo solicitamos la ejecución de la tarea **“uname”** para, a continuación, solicitamos el estado de la misma para ver que está en espera de asignación de recursos.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub sleep 10;bjobs ;sleep 5; bjobs;bjobs -w
Job <7223> is submitted to default queue <normal>.
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7223    rafa.el PEND  normal     hpc                    sleep 10   Sep  1 11:47
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7223    rafa.el RUN   normal     hpc         compute-5-1 sleep 10   Sep  1 11:47
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7223    rafa.electron RUN   normal     hpc         compute-5-19 sleep 10   Sep  1 11:47
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

La situación es similar a la anterior. Solicitamos la tarea **"sleep 10"** que no hace más que esperar 10 segundos. Inmediatamente solicitamos el estado todas las nuestras tareas, esperamos 5 segundos y volvemos a solicitar el estado de nuestras tareas por dos veces. En la primera de estas salidas vemos que nuestra tarea está en ejecución, aparentemente, en el nodo **"compute-5-1"**. Tras la segunda salida en la que hemos utilizado el parámetro **"-w"** podemos concluir que se está ejecutando en el nodo **"compute-5-19"**.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub sleep 10;bjobs ;sleep 5; bjobs -l
Job <7219> is submitted to default queue <normal>.
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7219    rafa.el PEND  normal     hpc                    sleep 10   Sep  1 11:25

Job Id <7219>, User <rafa.electron>, Project <default>, Status <RUN>, Queue <no
                     rmal>, Command <sleep 10>
Thu Sep  1 11:25:58: Submitted from host <hpc>, CWD <$HOME>;
Thu Sep  1 11:25:59: Started on <compute-5-15>, Execution Home </home/UCLM/rafa
                     .electron>, Execution CWD </home/UCLM/rafa.electron>;
Thu Sep  1 11:26:01: Resource usage collected.
                     MEM: 4 Mbytes;  SWAP: 58 Mbytes
                     PGID: 32086;  PIDs: 32086 32091


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

rafa.electron@hpc:/home/UCLM/rafa.electron#
```

En este caso tenemos un seguimiento detallado de la resolución de la tarea cuyo JOBID es el 7219. Podemos ver cuándo se lanzó y desde dónde, cuándo se asignó al nodo "compute-5-15" y los recursos reservados para su uso.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bjobs -l 7219

Job Id <7219>, User <rafa.electron>, Project <default>, Status <DONE>, Queue <n
                     ormal>, Command <sleep 10>
Thu Sep  1 11:25:58: Submitted from host <hpc>, CWD <$HOME>;
Thu Sep  1 11:25:59: Started on <compute-5-15>, Execution Home </home/UCLM/rafa
                     .electron>, Execution CWD </home/UCLM/rafa.electron>;
Thu Sep  1 11:26:11: Done successfully. The CPU time used is 0.0 seconds.

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

rafa.electron@hpc:/home/UCLM/rafa.electron#
```

Terminamos con la salida del proceso cuyo identificador es el 7219 y que ha finalizado. Además, nos indica cuándo terminó y si lo hizo de manera exitosa o con error.

Hay otros comandos dedicados a obtener información acerca del procesamiento de un determinado trabajo con el comando bhist, especialmente cuando esté finalizado.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bhist -l 7235

Job Id <7235>, User <rafa.electron>, Project <default>, Command <uname -a>
Thu Sep  1 13:49:54: Submitted from host <hpc> to Queue <normal>, CWD <$HOME>,
                     Output File <reg.out>;
Thu Sep  1 13:49:55: Dispatched to <compute-5-28>;
Thu Sep  1 13:49:55: Starting (Pid 19075);
Thu Sep  1 13:49:55: Running with execution home </home/UCLM/rafa.electron>, Ex
                     ecution CWD </home/UCLM/rafa.electron>, Execution Pid <190
                     75>;
Thu Sep  1 13:49:55: Done successfully. The CPU time used is 0.0 seconds;
Thu Sep  1 13:49:55: Post job process failed;

Summary of time in seconds spent in various states by  Thu Sep  1 13:49:55
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  1        0        0        0        0        0        1

rafa.electron@hpc:/home/UCLM/rafa.electron#
```

4.5 Lanzar un trabajo

A la hora de ejecutar los trabajos en cualquiera clúster del SSC se debe tener en cuenta que se ha de utilizar el gestor de colas de OpenLava, con lo que habrá que enviar los trabajos a la cola de ejecución utilizando el comando bsub.

Los parámetros más usuales del comando bsub los comentamos a continuación:

$ bsub -n <ncpu> -q <cola> -m <equipo> -J <nombre trabajo> -o <fichero> -e <fichero> <comando>

donde los parámetros tienen los siguientes significados:

Parámetro	**Significado**
-n <ncpu>	Número de slots a utilizar en el cálculo (Nº de cores)
-q <cola>	Cola que va a gestionar la ejecución de la tarea
-m <equipo>	Equipo/s en los que se solicita la ejecución de la tarea*
-J <nombre>	Nombre que le damos a la tarea
-o <fichero>	Fichero de salida de la tarea
-e <fichero>	Fichero de errores de la tarea
<comando>	Comando y parámetros de la tarea a ejecutar

* Ese equipo/s debe/n formar parte de la definición de la cola utilizada

Hay que remarcar que el comando bsub interpreta como comando de la tarea el primer parámetro que no vaya precedido de un guión "-". Así pues, el comando bsub interpretará como parámetros propios aquellos que aparezcan antes del comando y, evidentemente precedidos del citado guión. Todos aquellos elementos que aparezcan tras el comando serán interpretados como parámetro de la tarea a ejecutar.

Veamos algunos ejemplos de utilización del comando bsub.

El ejemplo de uso más sencillo consiste en solicitar la ejecución en el sistema de colas de un comando/programa. El sistema nos devuelve el JOBID de nuestra tarea.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub uname
Job <7214> is submitted to default queue <normal>.
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

En este otro ejemplo solicitamos la ejecución del comando “uname” utilizando la cola “normal”.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub -q normal uname
Job <7226> is submitted to queue <normal>.
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

Ahora solicitamos la ejecución del comando “uname” reservando 8 slots/cores para su ejecución. Esto no significa que la tarea se vaya a ejecutar utilizando los 8 slots reservados. Sólo que hemos reservados 8 slots. El tema del procesamiento en paralalelo será tratado en el apartado 5 con mayor profundidad.

```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub -n 8 sleep 10
Job <7236> is submitted to default queue <normal>.
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

Terminamos con un ejemplo solicitando la ejecución del comando “uname -a” enviando la salida al fichero reg.out

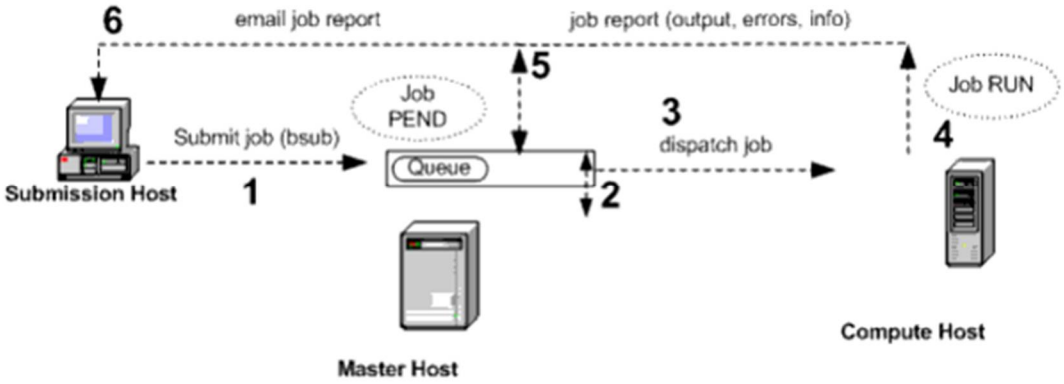
```
rafa.electron@hpc:/home/UCLM/rafa.electron# bsub -o reg.out uname -a
Job <7235> is submitted to default queue <normal>.
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

Podemos ver el contenido del fichero de salida con el comando “cat”

```
rafa.electron@hpc:/home/UCLM/rafa.electron# cat reg.out
Sender: OpenLava System <root@compute-5-28>
Subject: Job 7235: <uname -a> Done

Job <uname -a> was submitted from host <hpc> by user <rafa.electron>.
Job was executed on host(s) <compute-5-28>, in queue <normal>, as user <rafa.electron>.
</home/UCLM/rafa.electron> was used as the home directory.
</home/UCLM/rafa.electron> was used as the working directory.
Started at Thu Sep  1 13:49:55 2016
Results reported at Thu Sep  1 13:49:55 2016

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
uname -a "
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.01 sec.
    Max Memory :         2 MB
    Max Swap   :        29 MB

    Max Processes  :         1

The output (if any) follows:

Linux compute-5-28.ssc.uclm.es 2.6.32-573.22.1.el6.x86_64 #1 SMP Wed Mar 23 03:35:39 UTC 2016 x86_6
4 x86_64 x86_64 GNU/Linux
rafa.electron@hpc:/home/UCLM/rafa.electron#
```

4.6 Tabla de estados de un trabajo

A continuación, veremos un esquema representativo de los estados más significativos en los que se puede encontrar una tarea desde su generación hasta su finalización



La tarea se genera en un equipo servidor de acceso Submission host. Este equipo la envía al Master host. Se mantiene en estado PEND mientras que el Master host encuentra un equipo con recursos necesarios para poder ejecutar la tarea. Tras localizar dicho equipo que será un Compute host, se la remite. La tarea pasa a tener el estado RUN. Dicha ejecución probablemente genera una salida si todo va bien o, en otro caso, genera un informe de error. En cualquier caso, al finalizar la ejecución la tarea pasa a estado DONE y por correo electrónico se informa al usuario de la finalización.

Hay algunos estados más que aparecen en la siguiente tabla

Estado	**Descripción**
PEND	Tarea pendiente de ejecución
RUN	Tarea en ejecución
DONE	Tarea finalizada correctamente
EXIT	Tarea finalizada con algún error en el sistema de colas
PSUSP	Tarea suspendida por el usuario o el administrador mientras estaba en estado PEND
USUSP	Tarea suspendida por el usuario o el administrador mientras estaba en estado RUN
SSUSP	Tarea suspendida por el sistema OpenLava

5 Trabajos en paralelo

5.1 MPI

MPI (Message Passing Interface, Interface de paso de mensajes) es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores. El estándar final se presentó en la conferencia de Supercómputo en noviembre de 1993. (Wikipedia)

Algunas de las implementaciones relevantes que podemos citar son OpenMPI, Intel MPI, MPICH o MVAPICH. No es objetivo de este documento estudiar en profundidad las particularidades de

cada una de ellas y las diferencias entre ellas. Nos quedamos solo con los mecanismos para lanzar tareas con algunas de ellas, en concreto con las tres que hay instaladas en nuestros clústeres.

5.2 OpenMPI

Para acceder a la librería OpenMPI hay que tener configuradas convenientemente dos variables de entorno. Para ello basta añadir las siguientes dos líneas en el fichero $HOME/.bashrc

```
root@hpc:/root# more .bashrc
...
LD_LIBRARY_PATH=/usr/lib64/openmpi/lib:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH
PATH=/usr/lib64/openmpi/bin:$PATH;export PATH
```

De forma que las variables citadas tengan, al menos, los siguientes valores

```
root@hpc:/root# echo $LD_LIBRARY_PATH
/usr/lib64/openmpi/lib:
root@hpc:/root# echo $PATH
/usr/lib64/openmpi/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:
root@hpc:/root#
```

Esta configuración se puede conseguir bien con un procedimiento manual, bien ejecutando el comando

```
root@hpc:/root# configura ompi
root@hpc:/root#
```

y haciendo login de nuevo para cargar la nueva configuración

5.3 Intel MPI

Para acceder a la librería MPI de Intel hay que tener configuradas igualmente las variables de entorno para que apunten al repositorio correspondiente. La distribución de los compiladores de Intel nos proporciona scripts para activar la configuración necesaria.

```
root@hpc:/root# more .bashrc
...
source /opt/intel/bin/compilervars.sh intel64
source /opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/bin64/mpivars.sh intel64
export INTEL_LICENSE_FILE=/opt/intel/licenses/USE_SERVER.lic
```

De forma que las variables citadas tengan, al menos, los siguientes valores

```
rafa.electron@hpc:/home/UCLM/rafa.electron# echo $PATH
/opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/intel64/bin:/opt/intel/composer_xe_2013_sp1.2.144/bin/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mpirt/bin/intel64:/opt/intel/composer_xe_2013_sp1.2.144/debugger/gdb/intel64_mic/py26/bin:/opt/intel/composer_xe_2013_sp1.2.144/debugger/gdb/intel64/py26/bin:/opt/intel/composer_xe_2013_sp1.2.144/bin/intel64:/opt/intel/composer_xe_2013_sp1.2.144/bin/intel64_mic:/opt/intel/composer_xe_2013_sp1.2.144/debugger/gui/intel64:/opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/intel64/bin
rafa.electron@hpc:/home/UCLM/rafa.electron# echo $LD_LIBRARY_PATH
/opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/intel64/lib:/opt/intel/composer_xe_2013_sp1.2.144/compiler/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mpirt/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/ipp/../compiler/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/ipp/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/compiler/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mkl/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/tbb/lib/intel64/gcc4.4:/opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/intel64/lib:/opt/intel/compilers_and_libraries_2016.1.150/linux/mpi/mic/lib
```

Esta configuración se puede conseguir bien con un procedimiento manual, bien ejecutando el comando

root@hpc:/root# configura intel
root@hpc:/root# configura impi
root@hpc:/root#
y haciendo login de nuevo para cargar la nueva configuración

5.4 MPICH

Para acceder a la tercera librería instalada MPICH hay que tener correctamente configuradas dos variables de entorno. Para ello basta añadir las siguientes dos líneas en el fichero $HOME/.bashrc

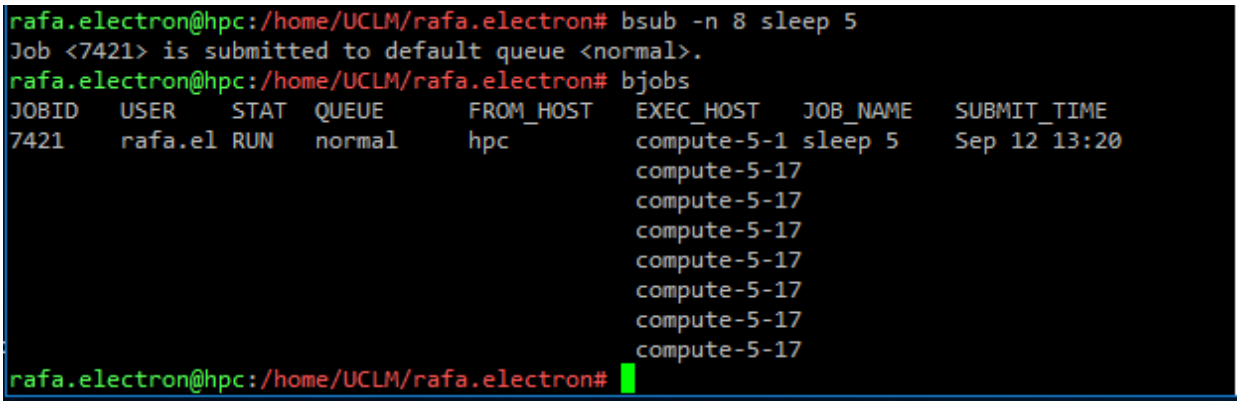
root@hpc:/root# more .bashrc
...
LD_LIBRARY_PATH=/usr/lib64/mpich/lib:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH
PATH=/usr/lib64/mpich/bin:$PATH;export PATH

De forma que las variables citadas tengan, al menos, los siguientes valores
root@hpc:/root# echo $LD_LIBRARY_PATH
/usr/lib64/mpichi/lib:
root@hpc:/root# echo $PATH
/usr/lib64/mpich/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:
root@hpc:/root#

Esta configuración se puede conseguir bien con un procedimiento manual, bien ejecutando el comando
root@hpc:/root# configura mpich
root@hpc:/root#
y haciendo login de nuevo para cargar la nueva configuración

5.5 Lanzar tareas en paralelo

Para lanzar tareas volvemos a utilizar el comando bsub con el parámetro –n para indicarle el número de slots que queremos reservar para nuestra tarea. Vemos un ejemplo elemental del comando utilizado para lanzar la tarea y del aspecto que presenta la tarea en ejecución.



Conviene destacar tras ver el ejemplo que el parámetro “-n” se utiliza para reservar slots de cálculo. El programa que nos interese debe ser capaz de aprovechar correctamente los slots reservados. En ese sentido, el ejemplo mostrado no es correcto. Se reservan 8 slots para una ejecución que solo utiliza uno.

El siguiente paso es lanzar trabajos paralelos. Esto se haría con el comando mpirun propio de la distribución de MPI que queramos usar indicando como parámetros el número de cores a usar, el conjunto de nodos sobre los que trabajar y el programa a ejecutar

```
rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi#  /usr/lib64/openmpi/bin/mpirun -n 12 -machinefile mpihosts.txt ./hello
Process 3 on compute-5-1.ssc.uclm.es out of 12
Process 0 on compute-5-1.ssc.uclm.es out of 12
Process 1 on compute-5-1.ssc.uclm.es out of 12
Process 2 on compute-5-1.ssc.uclm.es out of 12
Process 8 on compute-5-3.ssc.uclm.es out of 12
Process 9 on compute-5-3.ssc.uclm.es out of 12
Process 10 on compute-5-3.ssc.uclm.es out of 12
Process 11 on compute-5-3.ssc.uclm.es out of 12
Process 4 on compute-5-2.ssc.uclm.es out of 12
Process 5 on compute-5-2.ssc.uclm.es out of 12
Process 6 on compute-5-2.ssc.uclm.es out of 12
Process 7 on compute-5-2.ssc.uclm.es out of 12
rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi#
```

```
rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi# cat mpihosts.txt
compute-5-1 slots=4
compute-5-2 slots=4
compute-5-3 slots=4
compute-5-4 slots=4
compute-5-5 slots=4

rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi#
```

Hasta este momento hemos reservado slots para trabajo en paralelo y hemos lanzado trabajos en paralelo. Solo nos queda integrar todo esto con el sistema de colas con el que estamos trabajando. El problema aparece cuando, en lugar de proporcionar en la línea de comandos el nombre de los equipos a utilizar en la ejecución, corresponde al sistema de colas realizar de forma desatendida esa asignación.

Esto se consigue de forma automática haciendo uso de unos scripts preparados al uso y cuyo nombre es sobradamente significativo:

```
rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi# ls -la /opt/openlava/bin/*-mpirun
-rwxr-xr-x 1 root root 1491 Mar  8  2016 /opt/openlava/bin/intelmpi-mpirun
-rwxr-xr-x 1 root root 1367 Oct 22  2015 /opt/openlava/bin/lam-mpirun
-rwxr-xr-x 1 root root 1345 May 12 10:27 /opt/openlava/bin/mpich-mpirun
-rwxr-xr-x 1 root root 1387 Oct 22  2015 /opt/openlava/bin/openmpi-mpirun
rafa.electron@compute-5-1:/home/UCLM/rafa.electron/ompi#
```

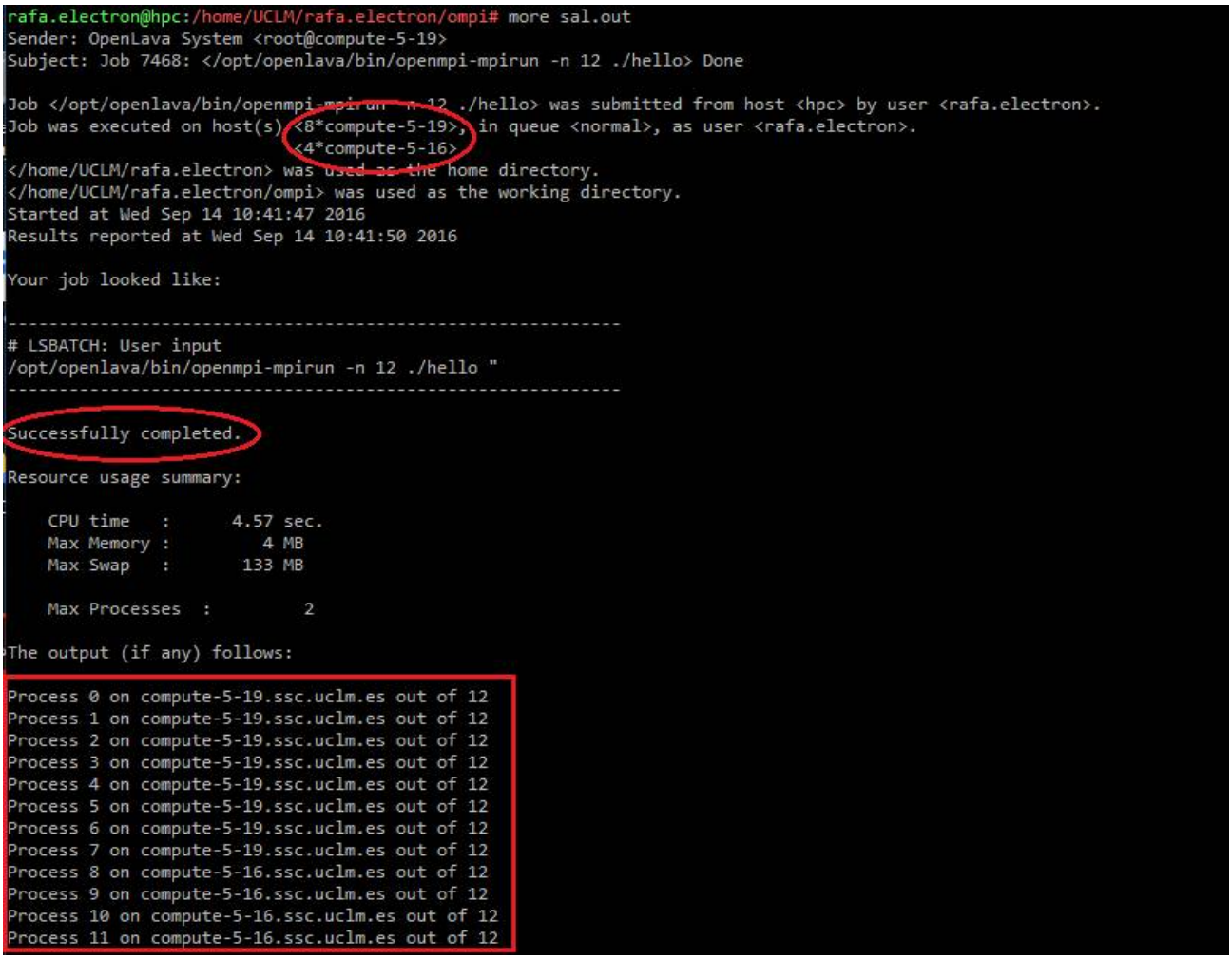
Para terminar con este bloque de ejemplos vamos a revisar un ejemplo del lanzamiento de una tarea completa usando el sistema de colas. El comando quedaría así. Hay que destacar que hacemos coincidir el nº de slots reservados con el nº de cores dedicados al proceso.

```
rafa.electron@hpc:/home/UCLM/rafa.electron/ompi# bsub -n 12 -e sal.err -o sal.out /opt/openlava/bin/openmpi-mpirun -n 12 ./hello
Job <7468> is submitted to default queue <normal>.
rafa.electron@hpc:/home/UCLM/rafa.electron/ompi#
```

Podemos ver que el trabajo se ha ejecutado en los nodos compute-5-19 y compute-5-16.

```
rafa.electron@hpc:/home/UCLM/rafa.electron/ompi# bjobs 7468
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
7468    rafa.el DONE  normal     hpc         compute-5-1 *2 ./hello Sep 14 10:41
                                             compute-5-19
                                             compute-5-19
                                             compute-5-19
                                             compute-5-19
                                             compute-5-19
                                             compute-5-19
                                             compute-5-19
                                             compute-5-16
                                             compute-5-16
                                             compute-5-16
                                             compute-5-16
rafa.electron@hpc:/home/UCLM/rafa.electron/ompi#
```

Por último, vemos la salida completa que nos ha generado la tarea.

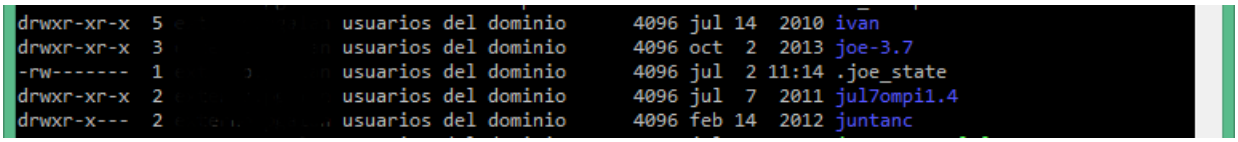


6 Buenas Prácticas

A continuación vamos a comentar algunas situaciones aparecidas en nuestros equipos y que utilizamos como ejemplos de lo que no se debe hacer para el correcto funcionamiento del sistema.

6.1 Instalación de software

En términos generales no es conveniente instalar software de uso generalizado en carpetas personales. En la siguiente imagen vemos la instalación del programa joe que es un sencillo editor de textos instalado en una carpeta de usuario.

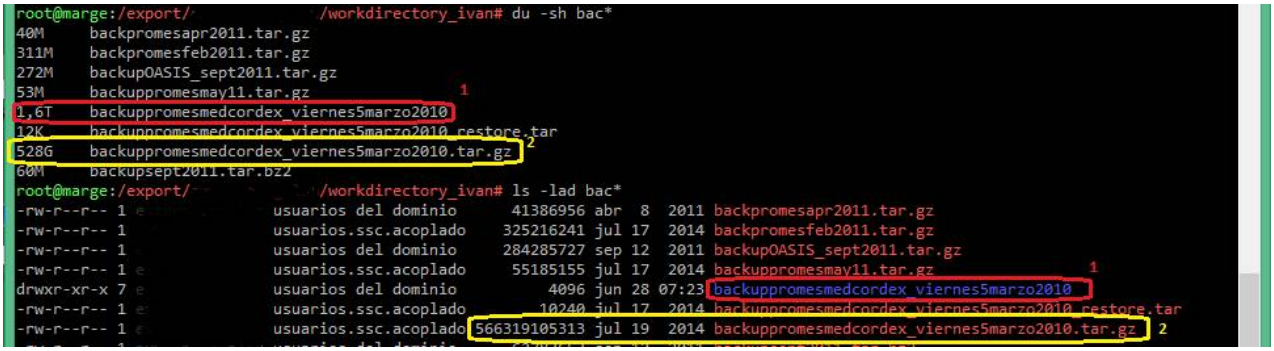


Se debería de haber solicitado mediante los procedimientos habituales una instalación general que habilite el uso de ese software a todos los usuarios con una única instalación.

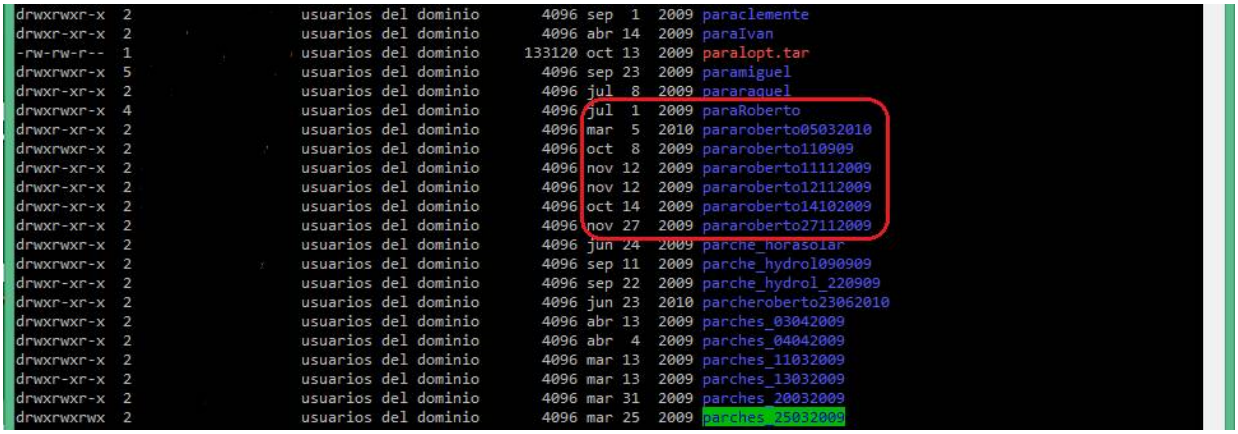
6.2 Servicio de repositorio

El SSC no es un servicio de almacenamiento ni un repositorio. El SSC proporciona servicio de almacenamiento en tanto que es imprescindible para cumplir con los objetivos de cálculo que sí son el objetivo fundamental del SSC. Vemos algunos casos a evitar en la medida de lo posible.

A continuación vemos una carpeta en la que se ha descargado un fichero comprimido bastante extenso y otra subcarpeta resultado de la descompresión del fichero comprimido anterior evidentemente bastante extensa también



Mantener un poco de orden resulta aconsejable. Existen mecanismos para controlar versiones que optimizan el uso de los recursos. Probablemente sea interesante no mantener el versionado que se realizó en 2009.



6.3 Invitados

En un sistema como el que nos ocupa es práctica habitual que tengamos colaboradores externos. De su correcta actuación son responsables sus respectivos anfitriones. Aquí vemos un invitado con una ocupación nada despreciable de 4,7 Terabytes.

6.4 Reserva de slots de cálculo

A la hora de reservar slots de cálculo con programas que son capaces de manejar varios slots en paralelo, es conveniente hacer coincidir la reserva de slots con la petición de uso. En la siguiente imagen de un fichero de comandos de Gaussian vemos como se configura una tarea para 8 slots.

```
root@hpcx:/home/UCLM/           /15-16/GQD/water# head G3water-em.com
nprocshared=8
%mem=12Gb
%chk=G3water-em.chk
#p opt td m062x/6-31g(d) scrf=(cpcm,solvent=water) int=ultrafine

Title Card Required
```

Sin embargo, en el momento de lanzar la tarea, se han reservado solamente 6 slots.

```
root@compute-3-14:/root# bjobs -u all -l 18629

Job <18629>, User <           >, Project <default>, Status <RUN>, Queue <normal
                     >, Command <g09 G3water-em.com>
Sun Nov 22 21:29:46: Submitted from host <hpcx>, CWD <$HOME/15-16/GQD/water>, 6
                     Processors Requested, Requested Resources <span[hosts=1]>;
```

Este ejemplo concreto hace que el equipo en el que se va a ejecutar la tarea tenga reservados 6 slots mientras que se ocupan 8 slots. En caso de recibir otra tarea, tendrá mayor oferta que recursos disponibles.

6.5 ¿Repositorio en la nube?

A cualquier usuario minimamente acostumbrado a trabajar con entornos Unix no le debe pasar por alto que la siguiente imagen corresponde a una carpeta copia la correspondiente carpeta de un equipo LinuX/Unix. Podríamos entender que, en sí, es una copia de seguridad de otro equipo.

```
root@lisa:/export/UCLM/             /usr# ls -la
total 32
drwxr-xr-x  8              usuarios del dominio 4096 jun  1  2013 .
drwx------ 55              usuarios del dominio 4096 nov  9 14:07 ..
drwxr-xr-x  2              usuarios del dominio 4096 nov 14  2013 bin
drwxr-xr-x  3              usuarios del dominio 4096 nov  9  2009 doc
drwxr-xr-x  3              usuarios del dominio 4096 nov  9  2009 include
drwxr-xr-x  2              usuarios del dominio 4096 nov 10  2009 lib
drwxr-xr-x  3              usuarios del dominio 4096 jun  1  2013 lib64
drwxr-xr-x  3              usuarios del dominio 4096 jun  1  2013 share
```

6.6 Cálculos en los servidores de acceso

Terminamos con una situación bastante más habitual de lo esperado. Debemos preservar los servidores de acceso de las tareas de cálculo. Esto es, los servidores de acceso realizan la función de recibir las conexiones de usuario. Si utilizamos estos servidores como sistemas de cálculo, podemos llegar a saturarlos y, en consecuencia, dejarán de atender correctamente las sesiones de conexión, tanto las establecidas como las nuevas. La cuestión es tan sencilla como no olvidar colocar el comando bsub, del que ya hemos hablado, delante de la tarea a realizar.

```
[rafa.electron@bart ~]$ R CMD BATCH prueba.R prueba.sal
[rafa.electron@bart ~]$
[rafa.electron@bart ~]$ bsub R CMD BATCH prueba.R prueba.sub
Job <27375> is submitted to default queue <qlarge>.
[rafa.electron@bart ~]$
```