Universitat Autònoma de Barcelona
Enginyeria Informàtica
Curs 2020-2021
Sistemas Distribuidos
Slurm en un entorno de Máquinas Virtuales

Datos

Profesor de Prácticas: **Sergio Villar, Carlos Montemuiño**
Tutorías:

Material de las prácticas

Para las prácticas en las máquinas Linux, la documentación será los *HOWTOs* y *manpages* del sistema. La presente guía es orientativa (no funcionará hacer un copy/paste sin saber lo que se hace).
Durante la realización de las prácticas se utilizarán máquinas virtuales (Virtual Machines VM) con OpenNebula. A OpenNebula se puede acceder desde todas las máquinas del laboratorio y desde fuera de la UAB. Existen varios templates con el sistema operativo Linux ya instalado.

Para acceder al entorno de prácticas se ha habilitado un acceso remoto a los servicios OpenNebula en esta dirección: http://nebulacaos.uab.cat:8443

La documentación básica sobre el entorno OpenNebula y cómo utilizarlo puede consultarse en el manual de operación: http://docs.opennebula.org/5.4/operation/index.html

Slurm[1] es un gestor de colas de trabajo de código abierto diseñado para *clusters* Linux de todos los tamaños. Proporciona tres funciones clave. En primer lugar se asigna acceso exclusivo y/o no exclusivo de los recursos (nodos de computación) para los usuarios, durante cierto tiempo, para que puedan realizar el trabajo. En segundo lugar, proporciona un marco para el inicio, ejecución y supervisión del trabajo en un conjunto de nodos asignados. Por último, se arbitra la contención de recursos mediante la gestión de una cola de espera de trabajo. La contención de recursos es crítica para ajustar las capacidades de un *cluster* (red, computación, almacenamiento) y el rendimiento (*throughput*) de una determinada aplicación. Así, Slurm ofrece herramientas para que el arquitecto/a de sistemas pueda evitar la sobrecarga de recursos (*stalled network, overswapping*, …) enviando a ejecución una carga de trabajo que pueda ser asumida por las capacidades hardware.

1. Descripción y configuración del entorno

Para la realización de esta práctica diseñaremos un entorno con cuatro VM: una máquina Master que contendrá el servidor Slurm, y tres máquinas Workers (*Worker[N]*) que ejecutarán los trabajos enviados al servidor Slurm. El entorno a crear se detalla en la figura 1.

La máquina *Master*, deberá tener una interfaz con acceso a Internet (*Interface Internet* con IP 10.10.10.0/24) y otra interfaz de red interna (*Interface Middle* con IP 20.20.20.0/24) que será la que comunicará con los *Worker*. En el caso de la máquinas *Worker*, deben tener una única interfaz con red interna (*Interface Middle* con IP 20.20.20.0/24), utilizada para comunicarse con el *Master* y entre máquinas *Workers*.

1 http://slurm.schedmd.com/

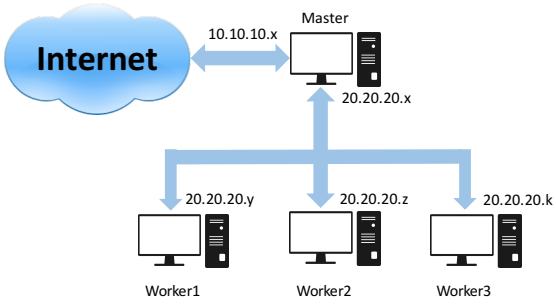

Figura 1: Entorno virtual utilizado en la práctica

Para la creación de las máquinas seguiremos el mismo procedimiento que el utilizado en la práctica 1 (apartado 2), teniendo que cuenta que para crear máquina Master, deberemos seleccionar la **imagen *SD-SLURM***, que contiene ya instalado el servidor Slurm, de esta forma no será necesario tener que instalarlo de cero, únicamente bastará con configurarlo. Como parámetros de configuración de la máquina *Master*, el nombre de la máquina virtual (*VN name*) será *master* (fijaos que está todo el nombre en minúscula), y no será necesario cambiar ningún otro parámetro. En cuanto al apartado *Networks*, se deben añadir dos interfaces de red. La primera interfaz debe ser conectada a la red con *Id 0* y *Name Internet*, mientras que la segunda interfaz se crea con *Id 1* y *Name Middle.* Una vez realizados estos cambios, crearemos la máquina *Master*.

En cuanto a las máquinas *Worker*, utilizaremos también la imagen ***SD-SLURM***, que puesto que ya trae instalado Slurm podremos ejecutar sus demonios en los *workers.* Como parámetros de configuración de la máquina *Worker*, el nombre de cada máquina virtual (*VN name*) será *worker[N]*, donde N será igual a 1, 2 y 3, quedando el nombre de las máquinas de la siguiente manera (worker1, worker2 y worker3). En cuanto al apartado Networks seleccionaremos la interfaz de red *Id 1* y *Name Middle.* Realizados estos cambios podremos proceder a crear las 3 máquinas virtuales.

Una vez hemos creado las máquinas y hemos accedido a ellas, será necesario cambiar el nombre de las máquinas virtuales dentro de la imagen, para ello modificaremos el fihero ***/etc/hostname*** y cambiaremos el nombre actual de las máquinas Workers (*localhost.localdomainworker)* por *worker[N]* (recordad que N= 1, 2, 3, dependiendo de la máquina). Puesto que la máquina Master ya ha sido configurada previamente, ya tiene el nombre master en ***/etc/hostname***, por lo que no será necesario modificarlo. Recordad que será necesario reiniciar las máquinas para que los cambios surtan efecto.

Además, en las cuatro máquinas se debe añadir al fichero ***/etc/hosts*** las IPs de las máquinas, quedando las primeras líneas del fichero similares a las de la figura 2. **Atención**: fijaos bien en las direcciones de vuestras máquinas para configurar bien vuestra red.

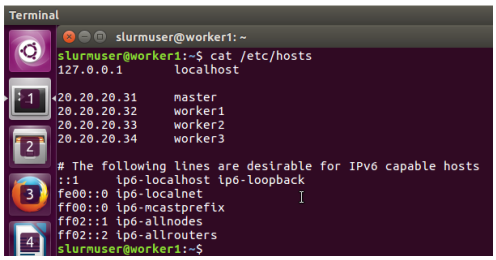

Figura 2: Ejemplo configuración /etc/hosts para la máquina *master*

2. Creación de usuario con sudo

A continuación, crearemos un usuario llamado *slurmuser* en cada máquina. El usuario *slurmuser* debe ser creado en las cuatro máquinas utilizando el comando *adduser*.

Una vez creados los usuarios, deberemos darle permisos de sudo. Para ello, modificaremos el fichero ***/etc/sudoers*** y añadiremos la siguiente línea:

alumno ALL=(ALL:ALL) ALL

Puesto que este procedimiento ya se realizó durante la primera práctica, no se entrará más en detalle, si necesita información más detallada, puede consultar el apartado 4 de la primera práctica.
Hay que tener en cuenta que cuando tengamos el entorno de master y worker[N] configurado, el usuario *slurmuser* deberá tener el mismo uid y gid en los ficheros ***/etc/passwd*** y ***/etc/group*** de todas las máquinas, para que SLURM trabaje correctamente. Es importante comprobarlo y en caso de tener uid o gid diferente, habremos de modificarlo a mano en los ficheros anteriores.

3. Configuración ssh

Antes de proceder a configurar el servidor Slurm, será necesario comprobar la conexión ssh entre las máquinas. SSH se encuentra instalado en las máquinas virtuales por lo que no hay necesidad de volverlo a instalar.

Primero se utiliza la máquina master, se cierra la sesión y se entra con el usuario slurmuser, con el cual probaremos la conexión con el worker1 utilizando el comando:

```
slurmuser@master:~$ ssh slurmuser@worker1
```

Ejecutado el comando la conexión debería fallar. Para realizar la conexión con éxito, es necesario cambiar el parámetro *PasswordAuthentication* del fichero ***/etc/ssh/sshd_config*** a “yes”.

Una vez hemos modificado el parámetro a yes, reiniciamos el servicio ssh de la siguiente manera:

```
slurmuser@master:~$ service sshd reaload
```

Realizaremos este procedimiento para el resto de máquinas. Una vez hecho esto para las cuatro máquinas, comprobamos la conexión ssh en cada una de las máquinas para asegurar que se comunican correctamente.

4. Autentificando usuarios Slurm

El siguiente paso sería instalar MUNGE[2] para autentificar usuarios de Slurm, crear la *munge key* e iniciar el servicio. Puesto que la máquina master ya tiene instalado el servidor Slurm, también tiene instalado MUNGE y creada la *munge key,* por lo cual no habremos de hacer nada en la máquina Master. La *munge key* la podemos encontrar en el siguiente fichero ***/etc/munge***, tal y como se muestra en el siguiente ejemplo:

```
slurmuser@master:~$ ls -l /etc/munge/
total 4
-r-------- 1 munge munge 1024 sep 28 11:54 munge.key
slurmuser@master:~$ cat munge.key
f7fbba6e0636f890e56fbbf3283e524c6fa3204ae298382d624741d0dc6638326e282c
41be5e4254d8820772c5518a2c5a8c0c7f7eda19594a7eb539453e1ed7
```

Puesto que las máquinas *Worker[N]* son una imagen de la máquina SD-SLURM, al igual que la máquina Master, también contienen instalado MUNGE y además poseen la misma *munge key*, por lo que no será necesario copiarla a las máquinas *Worker*. En el caso de que las máquinas *Worker[N]* no tuvieran la *munge key*, se debería copiar de la máquina *Master* a fin de que todas las máquinas que componen el sistema Slurm tuvieran la misma clave.

El siguiente paso será habilitar el servicio *munge* en el arranque del master y arrancar el servicio, para ello ejecutaremos los siguientes comandos:

```
slurmuser@master:~$ sudo systemctl enable munge
```

Arrancar el servicio de autenticación *munge* en el master:

```
slurmuser@master:~$ sudo systemctl start munge
```

Una vez hemos arrancado el servicio, comprobaremos que se puede generar una credencial *munge* y se pude decodificar localmente

```
slurmuser@master:~$  munge -n| unmunge
```

Repetimos estos tres pasos en las maquinas *worker[N]* para comprobar que se genera la credencial y se descodifica localmente.

Finalmente, para terminar este apartado comprobaremos si una credencial puede ser decodificada remotamente, usando el servicio ssh. Para ello, desde una máquina *worker[N],* ejecutamos el siguiente comando:

```
slurmuser@worker1:~$  munge -n| ssh master unmunge
```

En caso de haber tenido éxito, el *STATUS* de salida será ***Success***, tal y como se muestra en la siguiente imagen.

Figura 4: Decodificación remota de la credencial munge

5. Configurar *Slurm* y comprobar su funcionamiento

Configurar y administrar Slurm no es una tarea trivial en absoluto. No obstante, para nuestros propósitos podemos generar un fichero de configuración mínimo y suficiente, para ello abre el navegador y usa la herramienta web: http://slurm.schedmd.com/configurator.html con los siguientes valores:

master	**ControlMachine**
worker[1-3]	**NodeName**
root	**SlurmdUser**
/var/log/slurmctld.log	**SlurmctldLogFile**
/var/log/slurmd.log	**SlurmdLogFile**
/var/run/slurm-llnl/slurmctld.pid	**SlurmctldPidFile**
/var/run/slurm-llnl/slurmd.pid	**SlurmdPidFile**

Estos son los log del controlador del *master* y del demonio de los nodos

Deja el resto de valores por defecto. Verás que **SlurmdUser** no aparece en la plantilla web, por lo que habrá que descomentar la línea correspondiente una vez generado el *slurm.conf*. Para ello, dar al botón *submit* abajo del todo, copiar la configuración resultante a un nuevo fichero de configuración en ***/etc/slurm-llnl/slurm.conf*** . Editar el fichero y descomentar la linea *SlurmdUser=root*, igualmente hay que comentar el usuario SlurmUser.

Importante: antes de seguir adelante, comprueba que el campo *ControlMachine* aparece al principio del fichero ***slurm.conf y*** *tiene el valor master (ControlMachine=master)*, sino lo tenemos que incluir y comentamos el campo *SlurmctldHost=master* con un #, tal y como se muestra a continuación:

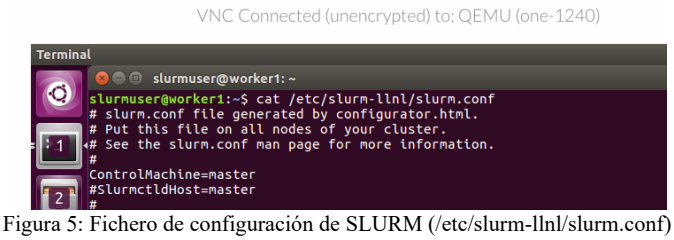

Figura 5: Fichero de configuración de SLURM (/etc/slurm-llnl/slurm.conf)

A continuación, nos aseguraremos que el fichero tenga los permisos adecuados:

```
slurmuser@master:~$ ls -l /etc/slurm-llnl/slurm.conf
-rw-r--r-- 1 root root 3192 sep 28 17:53 /etc/slurm-llnl/slurm.conf
```

Una vez hemos realizado estas comprobaciones, copiaremos el fichero *slurm.conf* de la máquina Master a los *worker[N]*. Es importante hacer la copia y no generar un archivos nuevos, ya que nos debe quedar exactamente igual en todas las máquinas, así evitaremos errores generando únicamente un único fichero de configuración. Por ello, lo generamos en la máquina Master y los copiamos a los *worker[N]* de la siguiente manera:

```
slurmuser@worker1:~$ scp slurmuser@master:/etc/slurm-llnl/slurm.conf .
slurmuser@worker1:~$ mv slurm.conf /etc/slurm-llnl/slurm.conf
```

Si la copia se ha realizado con éxito, nos mostrará un mensaje indicando que se ha transferido el 100% del fichero de la máquina Master a la máquina *worker[N]*, tal y como se indica en la figura 6, donde se copia el fichero de la máquina master a la máquina worker1:

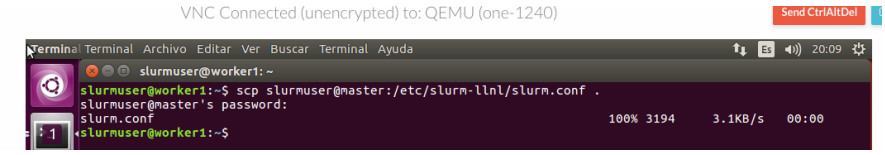

Figura 6: Transferencia del fichero slurm.conf de la máquina master a los workers

A continuación, procederemos a copiar el archivo *slurm.conf* al resto de máquinas workers.

Finalmente, arrancamos el controlador en el master y los demonios en los nodos *worker[N]*.

```
slurmuser@master:~$ sudo systemctl start slurmctld.service
slurmuser@master:~$ sudo systemctl status slurmctld.service
● slurmctld.service - Slurm controller daemon
   Active: active (running) since mar 2015-09-29 10:48:04 CEST; 9s ago
...
sep 29 10:48:04 master slurmctld[2654]: Running as primary controller
```

```
slurmuser@worker1:~$ sudo systemctl start slurmd.service
slurmuser@worker1:~$ sudo systemctl status slurmd.service
...
sep 28 18:26:01 worker1 systemd[1]: Started Slurm node daemon.
```

A ejecutar el comando *status* nos deberá aparecer en verde el texto *active (running)*, tal y como se muestra a continuación.

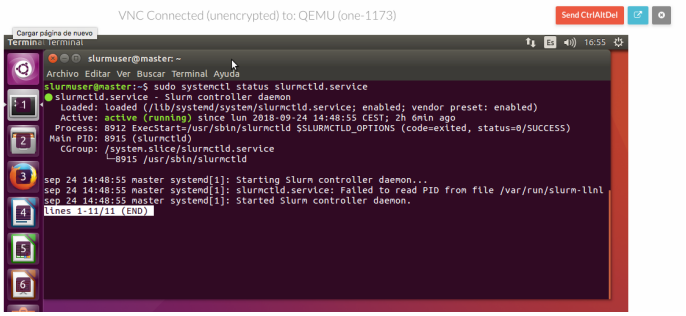

Figura 5: Servidor Slurm ejecutándose

Una vez tenemos ejecutando el controlador y los demonios, el siguiente paso es ejecutar un trabajo de prueba con el usuario *slumuser*. Para ello nos vamos a la máquina Master y ejecutamos el siguiente comando:

```
slurmuser@master:~$ srun -N1 /bin/hostname
worker1
```

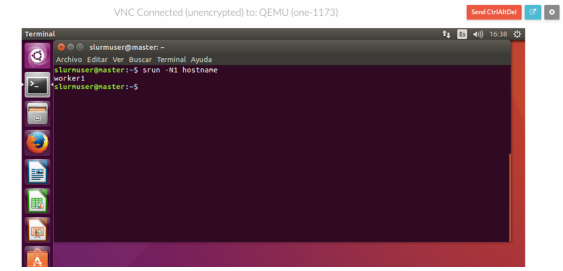

Figura 6: Salida del trabajo enviado

Como podemos ver, el trabajo se ha ejecutado con éxito en el worker1, ya que nos muestra por pantalla el *hostname* del worker1.

Recuerda gestionar los servicios *slurmctld* y *slurmd* con systemctl (status, start, stop) una vez terminado el trabajo con Slurm, para que no consumas recursos innecesarios.

6. Enviar y recuperar resultados de trabajos Slurm con srun

Slurm da soporte a distintas capacidades de ejecución de trabajos. Entre ellas destaca la contención de recursos, es decir, herramientas que permiten al usuario diseñar cargas de trabajo aptas a las capacidades (computación, almacenamiento, red) de los clusters. A continuación, realizaremos ejercicios que muestran algunas de las capacidades de Slurm, de manera muy resumida.

Slurm nos permite lanzar comandos desde cualquier nodo, en nuestro caso *master* o *worker[N]*, ya que todos ellos se conectan al controlador del *master*. Para ello, utilizamos el *slurmuser* con su munge key creada anteriormente.

El primer paso será comprobar que todos los nodos están activos, para ello utilizaremos el comando ***sinfo,*** el cual nos indica el estado de todos los nodos de cómputo que componen el sistema Slurm. El comando ***sinfo*** posee una gran variedad de parámetros los cuales permiten ordenar o filtrar nodos en función de su estado. Para los propósitos de esta práctica, ejecutaremos el comando sin parámetros, el cual nos muestra un resumen general de los nodos y su estado. Si todos los nodos aparecen como *idle*, significa que están activos y sin actividad, es decir, están libres para ser asignarles trabajos (*jobs*).

Comprueba que los nodos están levantados con el comando **sinfo.** Si todos los nodos están arrancados correctamente y preparados para trabajar, obtendremos una salida como la siguiente, donde los tres nodos están en estado *idle*:

```
slurmuser@worker2:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      3   idle worker[1-3]
```

En caso de que algún nodo estuviera en estado *down* (en lugar de idle), significa que el nodo no está activo, el tal caso deberemos consultar el log indicado para ver que está ocurriendo. Los logs los podemos encontrar en ***/var/log/slurmctld.log*** para el controlador del master, y en ***/var/log/slurmd.log*** para los demonios Slurm de los nodos workers. En caso de haber seguido seguido los pasos correctamente, todos los nodos nos deberían aparecer en estado *idle*.

Además del estado *idle* y *down,* otro posible estado es el *unknown*, este estado significa que el nodo está con su demonio *slurmd* activo, pero el controlador ha dejado de controlar como es debido. En tal caso, apagaremos todos los *worker[N]*, haciendo stop del demonio *slurmd* y los volveremos a arrancar.

```
slurmuser@worker1:~$ sudo systemctl stop slurmd.service
slurmuser@worker1:~$ sudo systemctl start slurmd.service
```

Una vez todos los nodos están en estado *idle*, lanzaremos un trabajo (job), con el comando ***srun***. El comando *srun* nos permite enviar un job a la cola de ejecución, el cual estará esperando hasta que haya nodos de cómputo disponibles para poder ser ejecutado, o ejecutarlo en tiempo real de manera interactiva. El comando *srun,* posee una amplia variedad de parámetros, los cuales permiten indicar los requerimientos que necesitará el job para ejecutarse, como podría ser: el número de mínimo y máximo de nodos que necesitará el trabajo, la cantidad máxima de memoria RAM necesaria, el tiempo máximo de ejecución, o la cantidad de espacio de disco, entre otras opciones.

El siguiente job se quedará ejecutando indefinidamente un simple *ping* (sin devolver el control al usuario, pues no termina nunca), reservando para su ejecución los tres nodos (*worker[N]*), aunque para un ping con un nodo sería suficiente. Como podemos ver en el ejemplo, el parámetro ***–Nx*** nos indica el número de nodos que queremos reservar.

```
slurmuser@master:~$ srun -N3 /bin/ping master
```

Desde otro terminal lanzaremos otro job con un simple *hostname*, el cual al no haber nodos libres se quedará en la cola (contención de recursos de CPU) sin devolver el control al usuario:

```
slurmuser@master:~$ srun -N3 hostname
```

Desde otro terminal comprobamos que los tres nodos están ocupados por un job (*alloc*):

```
slurmuser@master:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      3  alloc worker[1-3]
```

Como podemos intuir, el estado *alloc*, nos indica que el nodo se encuntra ocupado por otro job, y no está disponible para recibir nuevos trabajos.

Comprobamos la cola de trabajos de Slurm con el comando ***squeue***. Este comando nos muestra el estado del job o jobs, en caso de tener más de uno en ejecución, que se encuentran en la cola de slurm. Al igual que el resto de comandos, el comando *squeue* posee una amplia variedad de parámetros para ordenar o filtrar jobs. Es este caso, utilizaremos la opción *-u,* la cual nos permite filtrar por usuario, tal y como se muestra a continuación.

```
slurmuser@master:~$ squeue -u slurmuser
JOBID PARTITION    NAME     USER      ST TIME  NODES NODELIST(REASON)
16       debug              ping      slurmuser  R  4:54        3            worker[1-3]
```

El estado *R* nos indica que el job se está ejecutando, una vez terminado de ejecutar, si volvemos a ejecutar el commando *squeue*, veremos que no obtendremos nada, puesto que una vez ejecutado desaparece. En caso de tener un job esperando a ser ejecutado, nos aparece en estado *W (Waiting).*

Una vez realizada y comprobada la configuración de slurm, se realiza una serie de preguntas a fin de consolidar los conocimientos aprendidos en la práctica.

PREGUNTA 1: ¿En qué estado se encuentra el job?

Lanzamos otro job con un simple hostname:

```
slurmuser@master:~$ srun -N1 /bin/hostname
srun: job 17 queued and waiting for resources
```

PREGUNTA 2: ¿Devuelve el interprete de comandos el control al usuario que lanza el comando *srun*? ¿Por qué?

Cancelamos el job del *ping* con el comando **scancel**

El comando *scancel* permite detener "matar" de forma inmediata un job, para ello le pasaremos como parámetro el ID del job que queremos detener. Para conocer el ID de un job, ejecutaremos el comando *squeue.*

```
slurmuser@master:~$ scancel 16
```

PREGUNTA 3: En este punto ¿En qué estado quedan los nodos del *cluster* y por qué?

PREGUNTA 4: ¿Cuántos procesos *ping* se han ejecutado en nuestro ejemplo?

7. Envío asíncrono de Jobs y recuperación de resultados

En el apartado 7 hemos visto el comando *srun* para enviar jobs de manera síncrona (envía, ejecuta y cuando termina devuelve el control) esto no es muy operativo para la mayoría de los casos en los que tengamos multitud de trabajos a realizar. En este ejemplo vemos el comando *sbatch,* el cual envía trabajos al gestor de colas Slurm de manera asíncrona. Es decir, envía el trabajo, Slurm lo pone en la cola de trabajos, e inmediatamente devuelve el control al usuario. El usuario tendrá que monitorizar el estado de ejecución y cuando esté terminado, recuperar el resultado de la ejecución del trabajo.

Para probar el uso de *sbatch*, vamos a editar un script con parámetros Slurm para enviar a ejecutar a los nodos, para ello edita con *gedit* un fichero *parametric-random.bash* como este:

```
slurmuser@master:~$ cat parametric-random.bash
#!/bin/bash
#
#SBATCH -N 1 # number of nodes
#SBATCH -o slurm.%N.%j.out # STDOUT
#SBATCH -e slurm.%N.%j.err # STDERR

for i in {1..30000}
  do echo $RANDOM >> SomeRandomNumbers.txt
done

sort SomeRandomNumbers.txt
```

PREGUNTA 5 ¿Dónde volcará la salida el comando sort del script? ¿Hacia dónde redirecciona esa salida este script Slurm ? ¿Y la salida de errores (si los hubiese)?

PREGUNTA 6 ¿Qué significan las directivas Slurm de %N y %j ? ¿Para qué se utilizan en este script?

Envía el trabajo a ejecutar:

```
slurmuser@master:~$ sbatch parametric-random.bash
Submitted batch job 19
```

Monitoriza la cola de trabajos de ejecución con el comando *squeue* hasta que el trabajo haya terminado. Luego recupera la información generada como salida del trabajo.

Para recuperar la información, desde la máquina master utilizaremos el comando scp, el cual nos permite realizar transferencias de archivos entre máquina. IMPORTANTE: Fíjate bien en el ID de tu trabajo, al realizar la copia.

```
slurmuser@master:~$ scp worker1:~/slurm.worker*.19.* .
slurm.worker1.19.err          100%    0     0.0KB/s   00:00
slurm.worker1.19.out          100%  553KB 552.8KB/s   00:00
```

Formatted

PREGUNTA 7 ¿Qué contienen los ficheros de salida generados?

PREGUNTA 8: Copia el parametric-random.bash a parametric-random-V2.bash, y modifica esta segunda versión según estas características:

- Pon el número de nodos (-N) a 3
- Cada job generará 10000 números aleatorios.
- Sustituye las dos veces que aparece **SomeRandomNumbers.txt** por **SomeRandomNumbers.%j.txt** Con esto, el nodo que ejecute el trabajo crea un fichero SomeRandomNumber por cada trabajo (ya que todos los trabajos comparten el *home* de *slurmuser,* y si no distinguimos de alguna manera se mezclarán los resultados de un trabajo con otros)

Lanzamos con *sbatch* como antes, digamos que encola el job 30 y recuperamos el resultado, por ejemplo:

```
slurmuser@master:~$ scp worker1:~/slurm.worker*.30.*.out .
slurm.worker1.30.out          100%  553KB 552.8KB/s   00:00
```

Repite *scp* para *worker2* y *worker3*, pero sólo habrá un fichero .out
Si quieres comprueba el número de líneas.

```
slurmuser@master:~$ wc -l slurm.worker1.30.out
10000 slurm.worker1.30.out
```

PREGUNTA 9 ¿Cuántos procesos se han ejecutado? ¿Cómo se explica esto en comparación con la respuesta a **Pregunta 4**? Parece contradictorio, pero no lo es. Observa: copia un parametric-random-V3.bash y déjalo así:

```
slurmuser@master:~$ cat parametric-random-V3.bash
#!/bin/bash
#
#SBATCH -N 3 # number of nodes
#SBATCH -o slurm.%N.%j.out # STDOUT
#SBATCH -e slurm.%N.%j.err # STDERR
```

```
hostname
```

Ejecútalo con *sbatch* como antes y *scp* del fichero *.out*, vuelca su contenido con *cat* y después ejecuta:

```
slurmuser@master:~$ srun -N3 /bin/hostname
```

Ambos tienen -N3, que reservan 3 nodos para la ejecución del trabajo. Sin embargo, no dan la misma solución. Luego:

PREGUNTA 10 A partir de los resultados del problema ¿Cuál es la diferencia en la asignación de jobs a nodos enviando con *srun* y enviando con *sbatch*?