Universidad Industrial de Santander

Laboratorio de
Super Computación y Cálculo Científico
(SC3)
http://www.sc3.uis.edu.co

Gilberto Díaz
gilberto.diaz@uis.edu.co

Índice General

Capítulo 1

Introducción

Desde finales de los 60 Seymour Cray desarrolló computadores con capacidades mucho mayores que otros sistemas de la época. Durante los 70 fundó su propia compañía donde creó adelantos tecnológicos que hicieron que sus máquinas fueran mucho más rápidas. En los 90 estos adelantos fueron incorporados paulatinamente en los procesadores de los computadores personales. En 1994 Thomas Sterling y Donald Becker construyeron un súper computador virtual utilizando PCs cuya técnica denominaron **Clusters Beowulf**. Un *cluster* es una colección de computadores tradicionales interconectados por una red, y un conjunto de bibliotecas y aplicaciones que lo hacen ver como un único computador con mucho poder de cálculo [1].

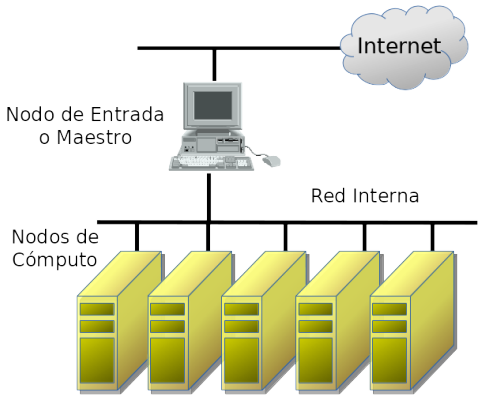

Figura 1.1: Arquitectura de un Cluster

En la década de los 2000 esta arquitectura fue incursionando seriamente en el mundo de la súper computación y para Junio de 2017 es la más utilizada.

Una de la actividades administrativas principales en un cluster es la gestión de los recursos. Esto se lleva a cabo por un *middleware* especializado que maneja la reservación

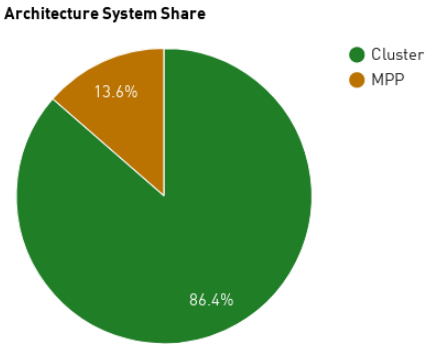

Figura 1.2: Arquitecturas top500 Junio 2017

de recursos para la ejecución de los programas, el balanceo de la carga de trabajo, etc. Desde los inicios de los clusters se han desarrollado varios sistemas para cumplir este objetivo. Actualmente uno de los más versátiles es SLURM.

SLURM (Simple Linux Utility for Resource Management) es un sistema de gestión de recursos de *clusters* de computación de alto rendimiento. Su desarrollo comenzó en el 2002 en colaboración entre el laboratorio nacional Lawrence Livermore, SchedMD,[2] Linux NetworX, Hewlett-Packard, and Groupe Bull [4]. Tiene alrededor de 500 mil líneas de código en lenguaje C. Es utilizado en más del 60 % de las máquinas de la lista del top500 [1]. Sus principales funciones son [5]:

- Asignar acceso exclusivo o compartido a los recursos.
- Proporcionar un ambiente de ejecución y supervisión de trabajos.
- Gestionar a través de una cola las solicitudes que superen los recursos disponibles.

1.1. Características

Las principales características de SLURM son:

- Código abierto: SLURM está desarrollado bajo la licencia GNU General Public License V2.
- Tolerancia a fallas: No tiene un único punto de falla, se puede contar con réplicas de servicios (demonios) para minimizar y proporciona opciones de tolerancia de fallas para los trabajos.

[1] http://www.top500.org

- Escalable: puede gestionar hasta cien mil trabajos independientes. Se puede utilizar para clusters pequeños o muy grandes.
- Altamente configurable: tiene más de 100 plugins para realizar varias funciones.
- Bases de datos integrada: utiliza MySQL para gestionar configuración y contabilidad.
- Programación de trabajos paralelos preferencial o grupal.
- Optimización de asignación de recursos inter nodos e intra nodos.
- Los nodos desocupados se pueden apagar.
- Los nodos pueden iniciar un sistema operativo distinto de acuerdo a los requisitos del trabajo.
- Programación de trabajos que utilicen recursos genéricos.
- Contabilidad en tiempo real a nivel de trabajo. Identifica trabajos con alto uso de CPU o memoria.
- Contabilidad para el uso energético por trabajo.
- Soporte para arreglos de trabajos.
- Soporte para MapReduce.

1.2. Arquitectura

SLURM cuenta con varios componentes para llevar a cabo sus funciones. La figura 5.3 ilustra los distintos elementos que lo conforman.

El demonio (**slurmctld**) proporciona el servicio principal el cual se encarga de gestionar los recursos y realizar la supervisión de los trabajos. Éste puede tener una instancia de respaldo que asuma las funciones en caso de fallas y se ejecuta en algunos de los nodos administrativos del cluster o en el nodo maestro (**frontend**). En los nodos de cómputo corre el servicio de gestión de ejecución de los trabajos (**slurmd**) que se encarga de esperar solicitudes, ejecutar el trabajo correspondiente a la solicitud y retornar el estado de cada trabajo. Así mismo, éste proporciona una comunicación jerárquica tolerante a fallas. Un servicio especial opcional que sirve para almacenar información del uso de los recursos catalogada por usuario (**slurmdbd**) y permite mantener información de varios clusters. Luego, se tiene el conjunto de herramientas administrativas que sirven para supervisar el sistema, modificar la configuración, etc. Finalmente, se tiene el conjunto de herramientas para gestionar los trabajos: iniciar, supervisar, terminar y mostrar el estado.

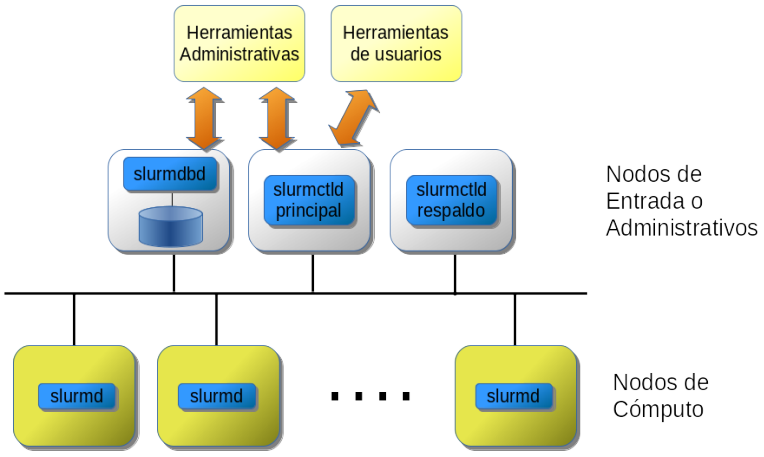

Figura 1.3: Arquitectura de SLURM

1.2.1. Organización de los Recursos

Las distintas entidades manejadas por los servicios de SLURM son:

Nodos

Los nodos constituyen los recursos de cómputo del sistema. Son los distintos computadores que realmente ejecutan los programas.

Particiones

SLURM tiene varios mecanismos para categorizar los recursos y así permitir una forma versátil de solicitar el uso de éstos. La primera forma de organizar los recursos es a través de particiones. Una partición es un mecanismo que permite al administrador agrupar de forma lógica los nodos de cómputo que tienen una característica en común y ser referenciados utilizando una sóla etiqueta. Por ejemplo, si se cuenta con un cluster heterogéneo, donde existe un grupo de nodos con más memoria que otros, se puede solicitar su uso a través del nombre de la partición. Una partición también puede ser considerada como una cola de trabajos con un conjunto propio de parámetros y valores tales como: tamaño límite del trabajo, tiempo límite del trabajo, usuarios permitidos, etc. Los recursos son asignados hasta que todos estén siendo utilizados. Las nuevas solicitudes deben esperar en una cola hasta que haya recursos suficientes que satisfagan la nueva solicitud.

Trabajos (Jobs)

Un trabajo o *job* es la asignación de recursos a un usuario específico por una cantidad de tiempo determinada. Los recursos asignados se deben especificar en la solicitud así como también el tiempo de uso.

Tareas (Job Steps)

Son el conjunto de programas (seriales o paralelas) que serán ejecutadas dentro de un trabajo.

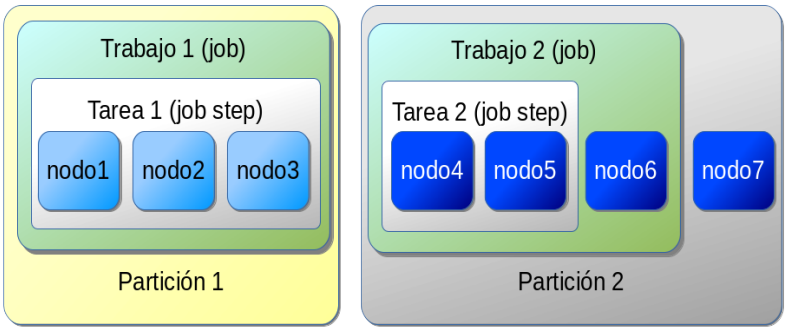

Figura 1.4: Particiones, jobs y jobs steps

Módulos (Plugins)

SLURM cuenta con un mecanismo general de gestión de módulos que permiten extender las funciones bajo un enfoque de bloques. Dentro de los módulos actuales se tiene:

- Almacenamiento y recopilación de contabilidad
- Recopilación de consumo energético.
- Autenticación de comunicaciones.
- Checkpoint
- Gestión de recursos genéricos.
- Envío de trabajos.
- Enlace con MPI.
- Gestión de prioridades.

- Selección de nodos.
- Calendarizador.
- Topología de red.

Capítulo 2

Configuración Preliminar

Para poder utilizar SLURM es necesario preparar el cluster para distribuir el sistema de archivos de los directorios hogar de los usuarios y el acceso a todos a todos los nodos para ejecutar los programas. Así mismo, es necesario hacer que todos los nodos conozcan los usuarios. Este capítulo muestra cómo realizar la configuración de estos aspectos.

2.1. Configuración de los Nombres de los Nodos del Cluster

Para facilitar la configuración de todos los servicios de un cluster es conveniente que estos puedan referenciarse a través de nombres. Hay varios mecanismos para realizar esto pero el más sencillo es construir un listado de todos los nodos con el siguiente formato y agregarlo al archivo **/etc/hosts**.

```
192.168.38.1  maestro
192.168.38.2  nodo1
192.168.38.3  nodo2
.
.
.
192.168.38.N  nodoN
```

Copie este archivo tanto al nodo maestro como los nodos de cómputo.

2.2. Configuración de Acceso Remoto

En un cluster los programas se ejecutan en los nodos de cómputo. Por lo tanto, es necesario permitir que éstos tengan permisos de ejecución remota de procesos. Actualmente,

esto generalmente se realiza a través del servicio **Secure Shell (SSH)**. Este servicio se basa en Infraestructura de Clave Pública para establecer un canal de comunicación seguro entre dos nodos de red. SSH es utilizado generalmente para abrir sesiones remotas o ejecución remota de comandos. De forma predefinida, ssh solicita una contraseña para permitir la comunicación, sin embargo, en un ambiente donde los programas se ejecutan mayoritariamente en segundo plano (**modo Batch**) la manejo de contraseñas impide tener un esquema de trabajo versátil. Por esto, cada usuario debe crear un par de llaves (pública y privada) para gestionar la comunicación de forma no interactiva. Para generar el par de llaves cada usuario debe ejecutar el siguiente comando:

```
ssh-keygen -f ~/.ssh/id_rsa -P ""
```

Este comando genera un conjunto de llaves pública y privada sin contraseña. Esto último se hace para agilizar la ejecución de los programas. Cree las llaves correspondientes a la cuenta de **root**

Para evitar que los usuarios tengan que lidiar con este servicio. Se puede utilizar un **script shell** que realice automáticamente todo el proceso la primera vez que el usuario ingrese al cluster. Asegúrese que el servio ssh esté disponible en cada nodo con el comando

```
ps ax | grep ssh
```

Si no está disponible, entonces instale e inicie el servicio con los siguientes comandos:

```
apt-get install openssh-server openssh-client
systemctl start sshd.service
```

Luego, reuna las identificaciones de cada nodo en un archivo texto utilizando el siguiente comando:

```
ssh-keyscan -t rsa nodo1,...,nodoN 2>/dev/null 1>/home/admin/known_hosts
```

Finalmente, cree un archivo (ssh.sh) en el directorio **/etc/profile.d** con el siguiente contenido:

```
if [ ! -d ~/.ssh ]; then
        sleep 2
        echo Bienvenido al Cluster Guane
        echo Es necesario configurar su ambiente SSH
        echo " "
        sleep 2
        echo Configurando el entorno SSH
        echo " "
        echo -n Generando llaves ssh.........................
        ssh-keygen -f ~/.ssh/id_rsa -P "" > /dev/null
        if [ $? -eq 0 ]; then
                echo -e "[\033[32m OK \033[30m]"
        else
                echo -e "[\033[31m FAIL \033[30m]"
                exit
        fi
        sleep 2
        echo -n Generando lista de nodos.....................
        cp /home/admin/known_hosts ~/.ssh/ &> /dev/null
        if [ $? -eq 0 ]
        then
                echo -e "[\033[32m OK \033[30m]"
        else
                echo -e "[\033[31m FAIL \033[30m]"
                exit
        fi
        sleep 2
        echo -n Generando acceso a los nodos ................
        cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys &> /dev/null
        if [ $? -eq 0 ]
        then
                echo -e "[\033[32m OK \033[30m]"
        else
                echo -e "[\033[31m FAIL \033[30m]"
                exit
        fi
        sleep 2
        echo -n Configurando permisos .......................
        chmod 644 ~/.ssh/authorized_keys ~/.ssh/known_hosts &> /dev/null
        if [ $? -eq 0 ]
        then
                echo -e "[\033[32m OK \033[30m]"
        else
                echo -e "[\033[31m FAIL \033[30m]"
                exit
        fi
        echo " "
        echo Configuracion finalizada!!
        echo " "
fi
```

2.3. Herramiena de Administración Paralela (C3 Tools)

Para una mejor administración del cluster es conveniente contar con una herramienta adecuada. **C3 Tools** permite ejecutar comandos en cada nodo de cómputo del cluster, copiar archivos a todos los nodos, etc. Para instalar C3 descargue los fuentes del sitio web

```
cd /opt
wget http://www.csm.ornl.gov/torc/C3/Software/5.1.3/c3-5.1.3.tar.gz
```

Luego, descomprima el archivo.

```
tar xvzf c3-5.1.3.tar.gz
```

Configure la aplicación creando un archivo (**/etc/c3.conf**) con el siguiente contenido

```
cluster nombreCluster {
        cluster:cluster    #Nombre del Nodo de Entrada
        nodo[1-5]          #Nodos de Cálculo
}
```

Agregue el camino donde están los ejecutables a la variable de ambiente **PATH** Ahora puede aprovechar las bondades de esta herramienta ejecutando comandos en todos los nodos de cómputo del cluster

```
cexec uptime
```

O puede compiar un archivo desde el nodo maestro a todos los nodos de cómputo

```
cpush /etc/profile.d/ssh.sh /etc/profile.d/
```

2.4. Configuración del Sistema de Archivos Distribuido

Existen muchos sistemas de archivos distribuidos que podemos utilizar para compartir los directorios hogares de los usuarios y otros directorios que se requieran. Aunque lo apropiado es contar con uno de altas prestaciones, para efectos de esta actividad seleccionamos uno de los más sencillos de utilizar: **Network File System (NFS)**. NFS funciona con un esquema cliente - servidor. El Servidor proporciona el espacio de almacenamiento y lo comparte a través de la red. Los clientes pueden tener acceso a ese espacio de almacenamiento de acuerdo a los permisos concedidos por el servidor. Para realizar este ejercicio instale el servidor en el nodo maestro y el cliente en todos los nodos de cómputo.

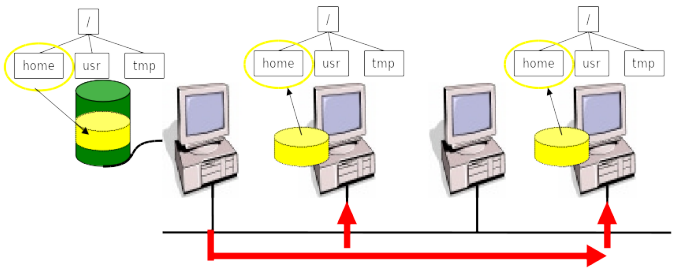

Figura 2.1: Network File System

2.4.1. Instalación de NFS

Con los siguientes comandos se instala el software necesario para activar el servicio de NFS. Ejecútelos en todos los nodos del cluster.

```
apt-get update
apt-get install nfs-common
```

Configuración del Servidor NFS

Edite el archivo **/etc/exports** y especifique los directorios que quiere compartir indicando cuales son los clientes y el tipo de acceso que se les garantizará

```
/home       nodo1(rw, no_root_squash, no_subtree_chec)
/home       pc1(rw,all_squash,anonuid=150,anongid=100)
/var/nfs    111.111.111.111(rw,sync,no_subtree_check)
/cdrom      *.dominio(ro)
```

Luego, inicie el servicio

```
service nfs-kernel-server start
```

Configuración del Cliente NFS

Para tener acceso local al sistema de archivos remoto ejecute el siguiente comandos

```
mount   servidor:/directorio_remoto   /directorio_local
```

Para hacer que el sistema de archivos esté disponible al momento de inicio de los clientes, ingrese la siguiente línea en el archivo **/etc/fstab**

```
maestro:/home   /home    nfs   rw,soft,bg,intr   0 0
```

Si existe una línea en ese archivo, entonces se puede montar el sistema de archivos con el siguiente comando

```
mount  /home
```

2.5. Configuración de Usuarios

En un cluster siempre es conveniente tener una base de datos centralizada de usuarios para facilitar la gestión de éstos. Se puede utilizar un servicio de directorio como OpenLDAP, sin embargo, la configuración de este tipo de servicios es un poco laboriosa. Para gestionar los usuarios en este taller lo haremos de forma manual, inscribiendo los usuarios en el nodo maestro y copiando los archivos de gestión de usuarios a todos los nodos de cómputo. Aprovecharemos C3 para realizar esta tarea

Ingrese como root en el nodo maestro y cree el usuario

```
useradd pepe
passwd pepe
New password: XXXX
Retype new password: XXXX
passwd: password updated successfully
```

Ahora copie los archivos respectivos a todos los nodos de cómputo

```
cpush /etc/passwd /etc
cpush /etc/shadow /etc
```

Con esto, todos los nodos conocerán a todos los usuarios que se creen en el nodo maestro.

Capítulo 3

Instalación de SLURM

Como se describió en la sección 1.2 SLURM tiene distintos componentes que funcionan en diferentes nodos, por lo tanto, la instalación debe realizarse en cada uno de ellos. Una forma de simplificar la instalación y la administración es compartir el directorio a través de un sistema de archivos distribuido. A continuación se describen los pasos a seguir para una instalación completa.

3.1. Instalar Requisitos

Instalar un sistema de autenticación tanto en los nodos de cómputo como en el nodo de maestro. En este caso usaremos **munge**. Utilice el sistema de gestión de paquetes para instalar **munge**. También, se necesita instalar algunos requisitos (comandos para Debian).

```
apt-get -y install munge libfreeipmi-dev libhwloc-dev freeipmi \
libmunge-dev
```

Cree la llave en el nodo de maestro.

```
/usr/sbin/create-munge-key
```

Copiar la llave a cada nodo.

```
scp /etc/munge/munge.key nodo1:/etc/munge/
scp /etc/munge/munge.key nodo2:/etc/munge/
...
scp /etc/munge/munge.key nodoN:/etc/munge/
```

Asegurarse que la llave en todos los nodos pertenezca al usuario *munge* y grupo *munge* y tenga los permisos correctos

```
ssh nodo1 chown munge:munge /etc/munge/munge.key
ssh nodo2 chown munge:munge /etc/munge/munge.key
...
ssh nodoN chown munge:munge /etc/munge/munge.key
```

Iniciar el servicio tanto en el nodo maestro como en los nodos de cómputo.

```
/etc/init.d/munge start
```

Probar munge. Desde el frontend ejecute los siguientes comandos
En el nodo maestro ejecute el siguiente comando

```
munge -n | unmunge
```

Desde el nodo maestro ejecute los siguientes comandos

```
munge -n | ssh nodo1 unmunge
munge -n | ssh nodo2 unmunge
```

El valor de STATUS debe ser Success

3.2. Descargar y Compilar SLURM

SLURM está disponible como paquete de la mayoría de las distribuciones populares. Sin embargo, aquí se prefiere compilar los fuentes para estandarizar el proceso de instalación sin importar el sabor de Linux que usted use.

Descargar slurm

```
cd /usr/local/src
wget https://www.schedmd.com/downloads/latest/slurm-17.02.7.tar.bz2
```

Compilar el programa

```
tar xvjf slurm-17.02.7.tar.bz2
cd /usr/local/src/slurm-17.02.7
./configure --prefix=/usr/local/slurm
make -j25
make install
```

Capítulo 4

Configuración de SLURM

La configuración de SLURM se realiza a través de archivos texto ubicados de generalmente el el subdirectorio **etc** del directorio de instalación. La explicación detallada de los campos utilizados en los archivos de configuración puede ver en los siguiente enlaces:

https://computing.llnl.gov/linux/slurm/slurm.conf.html
https://slurm.schedmd.com/documentation.html

4.1. Archivos de Configuración

A continuación se describen algunos de los archivos de configuración de SLURM. Todos estos archivos deben estar en todos los nodos del cluter. Cópielos en el directorio /usr/local/slurm/etc

1. El archivo **etc/allowed_devices.conf** contiene los *devices* que serán gestionados como recursos.

```
/dev/null
/dev/urandom
/dev/zero
/dev/cpu/*/*
/dev/pts/*
```

2. El archivo **etc/slurm.conf** contiene la configuración principal de SLURM. Para configuraciones sencillas este es el único archivo necesario.

```
ControlMachine=guane
MpiParams=ports=12000-12999
AuthType=auth/munge
CacheGroups=0
MailProg=/bin/mail
MpiDefault=none
ProctrackType=proctrack/cgroup
PropagateResourceLimitsExcept=MEMLOCK
ReturnToService=2
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=root
Slurmdlogfile=/var/log/slurm/slurmd.log
SlurmdDebug=7
Slurmctldlogfile=/var/log/slurm/slurmctld.log
SlurmctldDebug=7
StateSaveLocation=/var/spool
SwitchType=switch/none
TaskPlugin=task/cgroup
InactiveLimit=0
KillWait=30
MinJobAge=300
SlurmctldTimeout=120
SlurmdTimeout=10
Waittime=0
FastSchedule=1
SchedulerType=sched/backfill
SchedulerPort=7321
SelectType=select/cons_res
SelectTypeParameters=CR_Core,CR_Core_Default_Dist_Block
AccountingStoreJobComment=YES
ClusterName=guane
JobCompType=jobcomp/none
JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/linux
GresTypes=gpu

NodeName=guane[01-05] Procs=16 Sockets=2 CoresPerSocket=4 ThreadsPerCore=2
RealMemory=102000 Gres=gpu:8 State=UNKNOWN

NodeName=guane[06-16] Procs=24 Sockets=2 CoresPerSocket=6 ThreadsPerCore=2
RealMemory=102000 Gres=gpu:8 State=UNKNOWN

PartitionName=all Nodes=guane[01-16]   State=UP Default=YES Shared=YES
PartitionName=manycores16 Nodes=guane[01-05]  State=UP Shared=YES
PartitionName=manycores24 Nodes=guane[06-16]   State=UP Shared=YES
```

3. El archivo **etc/cgroup.conf** tiene la configuración de la gestión de los recursos

```
CgroupAutomount=yes
CgroupReleaseAgentDir="/usr/local/slurm/etc/cgroup"
ConstrainCores=yes
TaskAffinity=yes
ConstrainDevices=yes
AllowedDevicesFile="/usr/local/slurm/etc/allowed_devices.conf"
ConstrainRAMSpace=no
```

4. Copiar el script para liberar recursos de cpuset

```
mkdir etc/cgroup
cp /usr/local/src/slurm-14.11.7/etc/cgroup.release_common.example \
/usr/local/slurm/etc/cgroup/cgroup.release_common

ln -s /usr/local/slurm/etc/cgroup/cgroup.release_common \
/usr/local/slurm/etc/cgroup/release_devices

ln -s /usr/local/slurm/etc/cgroup/cgroup.release_common \
/usr/local/slurm/etc/cgroup/release_cpuset

ln -s /usr/local/slurm/etc/cgroup/cgroup.release_common \
/usr/local/slurm/etc/cgroup/release_freezer
```

4.2. Iniciar los Servicios

Se debe iniciar los distintos demonios de SLURM de acuerdo al tipo de nodo. En el frontend se incia el demonio **slurmctld** mientras que en los nodos **slurmd**.

Ejecute desde el frontend el siguiente comando:

```
cexec /usr/local/slurm/sbin/slurmd -M
```

Ejecutar en el frontend los siguientes comandos

```
slurmctld
scontrol update NodeName=guane[1-10] State=RESUME
```

4.3. Comprobación del Sistema

Revisar que los nodos esten online y correctamente configurados.

```
scontrol show node
sinfo
```

Ejecute el comando *sinfo*. El estado de los nodos de cómputo debe aparecer como **idle**.

Para probar que SLURM está funcionando ejecute un trabajo sencillo con el siguiente comando:

```
srun -N2 /bin/hostname
```

Capítulo 5

Administración de SLURM

5.1. Supervisión

Una de la tareas más importantes en la administración de SLURM es conocer el estado de todos los componentes y servicios. Esta sección muestra las herramientas que nos ayudan en la supervisión de SLURM

5.1.1. sinfo

Este comando muestra el estado de los distintos nodos. Un ejemplo de la salida es la siguiente:

```
sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
all*           up   infinite      2  down* guane[02,10]
all*           up   infinite      7    mix guane[04,07,11,13-16]
all*           up   infinite      1  alloc guane03
all*           up   infinite      6   idle guane[01,05-06,08-09,12]
manycores16    up   infinite      1    mix guane07
manycores16    up   infinite      1  alloc guane03
manycores16    up   infinite      3   idle guane[05-06,08]
manycores24    up   infinite      2  down* guane[02,10]
manycores24    up   infinite      6    mix guane[04,11,13-16]
manycores24    up   infinite      3   idle guane[01,09,12]
```

5.1.2. scontrol

Además de mostrar la configuración actual de SLURM, este comando es utilizado para modificarla, esto incluye elementos como trabajos (jobs), tareas (job steps), nodos, particiones, reservaciones, etc. Debe ser ejecutado por *root*. Si no se le pasan opciones se inicia

una línea de comandos desde donde se puede utilizar todas las opciones de forma interactiva. Si se realiza una modificación a la configuración general, se puede escribir la nueva configuración con el comando *write*, lo cual generará un nuevo archivo (slurm.conf.fecha) en el directorio de la actual configuración.

```
scontrol show nodes

NodeName=guane01 Arch=x86_64 CoresPerSocket=6
   CPUAlloc=0 CPUErr=0 CPUTot=24 CPULoad=0.08 Features=(null)
   Gres=gpu:8
   NodeAddr=guane01 NodeHostName=guane01 Version=14.11
   OS=Linux RealMemory=102000 AllocMem=0 Sockets=2 Boards=1
   State=IDLE ThreadsPerCore=2 TmpDisk=0 Weight=1
   BootTime=2017-08-26T16:36:30 SlurmdStartTime=2017-08-28T09:38:41
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

5.1.3. squeue

Este comando sirve para mostrar el estado de los trabajos.

```
squeue

JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
14842       all md_mpi.j jpvillab  R 1-12:00:06      1 guane07
14862       all openFoam edxon.me  R    9:15:34      2 guane[03-04]
14866       all OpenFOAM   irueda  R    8:33:04      4 guane[13-16]
```

Se puede utilizar una salida con formato para visualizar información específica

```
squeue -o "%u   %N      %b"

USER                    NODELIST        GRES
jpvillabonamonsalve     guane07         (null)
edxon.meneses           guane[03-04]    (null)
irueda                  guane[13-16]    (null)
```

Se puede obtener información con formato sobre un trabajo específico

```
squeue -n openFoam -o "%u %N"

USER       NODELIST
irueda     guane[13-16]
```

5.1.4. smap

Esta herramienta sirve para visualizar gráficamente información sobre trabajos, particiones y otros parámetros de la configuración

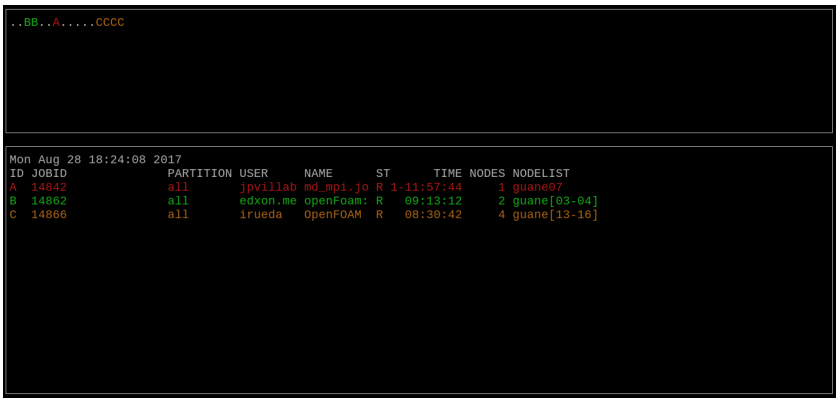

Figura 5.1: Herramienta smap

5.1.5. sview

Interfaz gráfica para visualizar y modificar el estado de SLURM

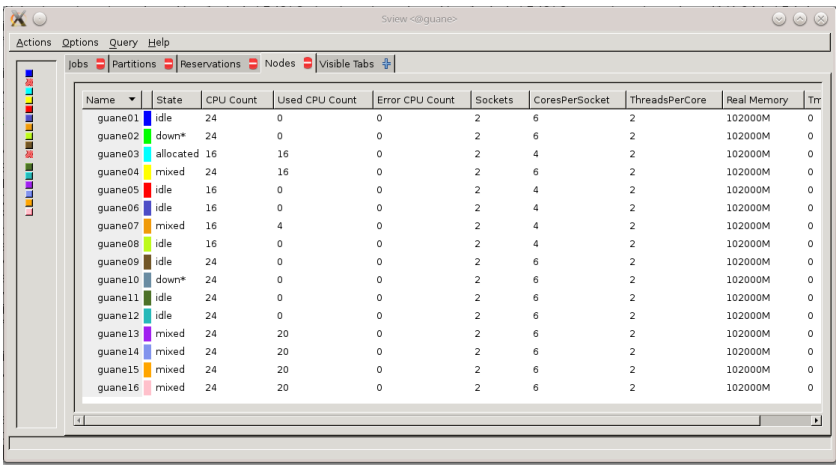

Figura 5.2: Herramienta sview

5.1.6. webslurm

Esta es una herramienta de supervisión web de SLURM. Se puede mostrar los trabajos en cola, las particiones, las reservaciones y mapas de trabajos de acuerdo a los cores disponibles organizados por rack.

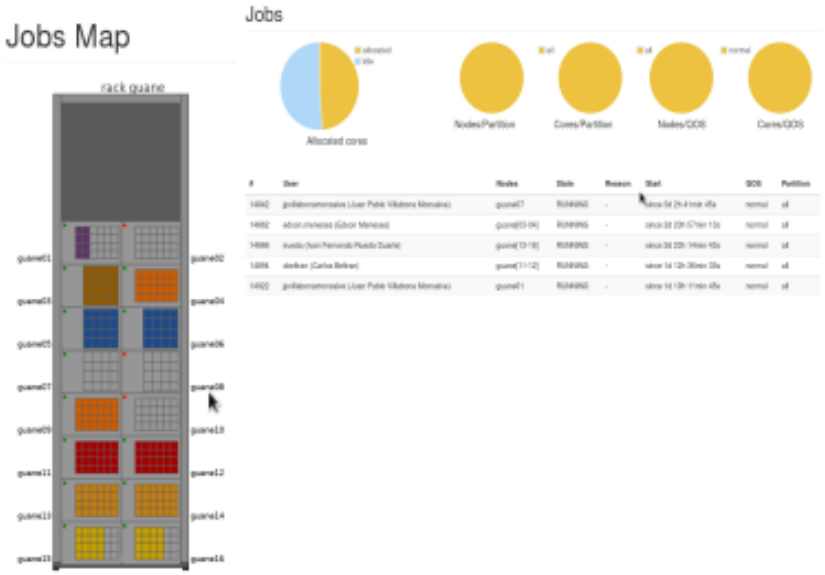

Figura 5.3: Herramienta webslurm

5.2. Resolución de Problemas

5.2.1. Depuración

En caso de presentarse algún inconveniente es bueno ejecutar el demonio principal en modo de depuración. Esto se hace con el siguiente comando:

```
slurmctld -Dvvvvvvvvvvvvv
```

5.2.2. Archivos de Registros

El administrador puede utilizar la información que se almacena en los archivos de registros (**logs**). En estos se puede encontrar registros que pueden indicar algún problema en los distintos componentes de SLURM. En el caso de este manual, los archivos de logs de los servicios principales son:

```
/var/log/slurm/slurmctld.log
/var/log/slurm/slurmdbd.log
```

Y el de los nodos de cómputo son:

```
/var/log/slurm/slurmd.log
```

Capítulo 6

Uso de SLURM

En este capítulo se describen los detalles de cómo utilizar SLURM desde el punto de vista del usuario. Se muestran las distintas herramientas y sus opciones

6.1. Ejecución de Programas Utilizando SLURM

Existen dos herramientas esenciales para la ejecución de trabajos

- **srun** Este comando ejecuta trabajos paralelos en el cluster. Si es necesario, srun solicitará recursos antes de la ejecución del trabajo.
- **sbatch** Este comando ejecuta un **script bash** en SLURM

6.1.1. ¿Cómo Ejecutar Trabajos Seriales?

Para ejecutar programas seriales pasivos, es decir, que no requieran de alguna entrada de parámetros de forma interactiva, ejecute un comando como el que sigue:

```
srun ./programa &> salida &
```

Este comando ejecuta **./programa** en alguno de los nodos del cluster. El argumento &> salida indica que tanto la salida como los errores se almacenan en un archivo de nombre salida que se creará en el mismo directorio desde donde se ejecutó el programa. Finalmente, el últmo caracter & es para liberar la cónsola desde donde se lanza la ejecución del programa.

6.1.2. ¿Cómo Ejecutar Trabajos Seriales Usando un Script Shell?

Para ejecutar programas seriales pasivos utilizando un script shell, que no requieran de alguna entrada de parámetros de forma interactiva, ejecute un comando como el que sigue:

```
sbatch ./script.sh
```

Este comando ejecuta **./script.sh** en alguno de los nodos del cluster.

6.1.3. ¿Cómo Ejecutar Trabajos Paralelos con OpenMP? (Memoria Compartida)

Para ejecutar programas paralelos con OpenMP edite un script shell de nombre **openmp.sh** con el siguiente contenido:

```
#!/bin/bash
export OMP_NUM_THREADS=4
./programa_ejecutable > salida
```

La línea **export OMP_NUM_THREADS=4** indica que se utilizarán 4 hilos de ejecución. El valor de esta variable debe fijarse según las necesidades y características de su programa ejecutable (**programa_ejecutable**). Luego, ejecute siguiente comando:

```
sbatch ./script.sh
```

Este comando ejecuta el script **./openmp.sh** en alguno de los nodos del cluster.

6.1.4. ¿Cómo Ejecutar Trabajos Paralelos con OpenMPI? (Memoria Distribuida)

Para ejecutar programas paralelos con OpenMPI edite un script shell, de nombre **mpi.sh**, con el siguiente contenido:

```
#!/bin/bash
#SBATCH --partition=all       #Seleccione los nodos para el trabajo de todos el
      #conjunto de nodos de cómputo del cluster
#SBATCH -o mpi.%j.out         #Nombre del archivo de salida
#SBATCH -J mpiJob             #Nombre del trabajo
#SBATCH --nodes=2             #Numero de nodos para correr el trabajo
#SBATCH --ntasks=20           #Numero de procesos
#SBATCH --tasks-per-node=10   #Numero de tareas por nodo

#Prepara el ambiente de trabajo
export I_MPI_PMI_LIBRARY=/usr/local/slurm/lib/libpmi.so
ulimit -l unlimited

#Ejecuta el programa paralelo
srun  ./programaMPIejecutable
```

Las líneas que comienzan con la palabra reservada **#SBATCH** indican las opciones del trabajo paralelo. La opción –**partition=all** indica que los nodos necesarios para ejecutar el trabajo se seleccionarán de la partición de nombre **all**.

La opción –**nodes=2** solicita 2 nodos de cómputo para el trabajo, la opción –**ntasks=20** indica que se ejecutarán 20 tareas que en el caso de MPI representan los procesos. La opción –**ntasks-per-node=10** indica que los 20 procesos se distribuirán de a 10 procesos por nodo. La siguiente línea enlaza la biblioteca con la que se compila OpenMPI para que tenga soporte integrado con SLURM

```
export I_MPI_PMI_LIBRARY=/usr/local/slurm/lib/libpmi.so
```

Y la siguiente línea es para eliminar el límite de memoria que el sistema opertativo fija de forma predefinida.

```
export I_MPI_PMI_LIBRARY=/usr/local/slurm/lib/libpmi.so
```

Finalmente, en la última línea, se especifica el programa ejecutable (**programaMPIejecutable**).

Luego, ejecute siguiente comando:

```
sbatch ./mpi.sh
```

Este comando ejecuta el script **mpi.sh** en el conjunto de los nodos del cluster.

6.1.5. Crear una Sesión Interactiva

SLURM permite conectarse a cualquiera de los nodos de cómputo y abrir una línea de comandos completamente funcional. Generalmente esto se hace para realizar pruebas preliminares de los programas que serán ejecutados en segundo plano (batch).

```
salloc srun --pty bash
```

Capítulo 7

Contabilidad

Registrar el uso de los recursos es fundamental para llevar estadísticas, prever crecimiento y establecer políticas. Este capítulo muestra como configurar el sistema de contabilidad de SLURM.

7.1. Configuración del Servicio Principal

Agregue las siguientes líneas en el archivo **slurm.conf**

```
AccountingStorageEnforce=associations,limits
AccountingStorageHost=localhost
AccountingStorageLoc=slurmAcctDB
AccountingStoragePort=6819
AccountingStorageType=accounting_storage/slurmdbd
```

7.2. Configuración del Servicio de Base de Datos

El archivo **etc/slurmdbd.conf** contiene los parámetros de gestión de la base de datos. Cree un archivo con este nombre en el directorio etc de SLURM con el siguiente contenido

```
AuthType=auth/munge
DbdAddr=localhost
DbdHost=localhost
DbdPort=6819
DebugLevel=4
LogFile=/var/log/slurm/slurmdbd.log
PidFile=/var/run/slurmdbd.pid
StorageType=accounting_storage/mysql
StorageHost=localhost
StoragePass=xxxxxx
StorageUser=slurmacct
StorageLoc=slurmAcctDB
```

Para efectos de recolección de información de uso de recursos SLURM utiliza una base de datos. Se debe crear el usuario de la base de datos con los permisos apropiados. La base de datos de este ejemplo residirá en el nodo de entrada.

```
mysql -u root -p
mysql> GRANT ALL  ON slurmAcctDB.* to 'slurm'@'localhost';
mysql> exit
```

7.3. Iniciar el Servicio

Inicie el servicio de contabilidad con el siguiente comando

```
/usr/local/slurm/sbin/slurmdbd &
```

7.4. Crear el Esquema de Contabilidad

Se debe crear al menos un esquema para recolectar la información del uso de los recursos. El siguiente ejemplo crea un esquema general. Usted puede crear los esquemas que necesite y utilizarlos para organizar los usuarios. Ejecute los siguientes comandos:

```
sacctmgr add cluster guane
sacctmgr add account general Description="General Accounting" Organization=SC3
```

7.5. Agregar los Usuarios al Esquema

Para que un usuario pueda utilizar los recursos, una vez que se activa la contabilidad, debe ser agregado a un esquema o categoría de la base de datos con el siguiente comando:

```
sacctmgr add user gilberto DefaultAccount=general
```

7.6. Visualización de la Contabilidad

Mostrar los usuarios y sus asociaciones

```
sacctmgr show associations
```

Para ver las estadísticas de la contabilidad utilice el comando **sacct**. Ejemplo:

```
sacct --format=user,totalcpu,alloccpus,nnodes,ncpus,jobid,state
```

Capítulo 8

Programación de Trabajos (Scheduling)

8.1. Tipos de Programadores (Schedulers)

Para el despacho óptimo de trabajos SLURM utiliza dos mecanismos:

- Un algoritmo basado en la curva de Hilbert.
- Una topología de red de árbol ancho.

El parámetro **SchedulerType** especifica el plugin del scheduler a utilizar.
Existen varios plugíns para planificar los trabajos, dentro de los nativos están:

- sched/builtin - FIFO
- sched/backfill - Relleno
- sched/gang - Tiempo compartido

Los plugins externos que se pueden utilizar son:

- sched/wiki - Maui
- sched/wiki2 - Moab

Los parámetros que controlan la programación de los trabajos son:

- **default_queue_depth=#** Especifica el número de trabajos a considerar por el planificador en cada evento (el valor predefinido es 100 trabajos)
- **defer** No se realiza el intento de planificar trabajos en el tiempo de despacho (submit).
- **max_switch_wait=#** Especifica el tiempo máximo que un trabajo puede esperar por el número deseado de leaf switches (el valor predefinido son 300 segundos)

8.2. Prioridades

SLURM puede manejar distintos modos de prioridades para el despacho de trabajos.

- **Priority/basic** FIFO
- **Priority/multifactor** Establece la prioridad de un trabajo de acuerdo a la suma de los pesos de una serie de factores [2].
- **Priority/multifactor2** Es una variación del anterior basada en tickets

La prioridad **P** de **multifactor** es una función como sigue:

$$P = Wedad \times Vedad + Wfair \times Vfair + Wtam \times Vtam + Wpart \times Vpart + Wqos \times Vqos \quad (8.1)$$

Donde:

- Los factores **W** son enteros de 32 bits
- Los factores **V** son punto flotante en el rango [0.0 - 1.0]
- **fair** es la política fairshare
- **tam** es el tamaño del trabajo
- **part** es la partición solicitada por el trabajo
- **qos** es la regla de Qos solicitada por el trabajo

En general, mientras más permanezca un trabajo en la cola esperando por recursos, aumenta su edad. **PriorityMaxAge** define el tiempo máximo en que la edad llega a su máximo, el valor predefinido es 7 días. **PriorityWeightAge** es un entero que se utiliza para acelerar la contribución del factor edad. De la misma manera, otros parámetros funcionan para el resto de los factores: **PriorityWeightPartition, PriorityWeightQOS**

Respecto a la prioridad de la partición, el parámetro **PriorityWeightPartition** controla el peso. Mientras más grande sea el valor, más grande es la prioridad de los trabajos dentro de la partición.

Cada regla de calidad de servicio también puede tener su aporte a la prioridad del trabajo. Esto se hace con el parámetro **PriorityWeightQOS**.

Uno puede hacer que SLURM considere el tamaño del trabajo en el cálculo de la prioridad, favoreciendo a los trabajos grandes o a los pequeños. De forma predefinida, mientras más grande sea el tamaño mayor será el factor correspondiente. Los parámetros que controlan este factor son:

- **PriorityFavorSmall=NO** Si se fija este parámetro en **NO**, aquel trabajo que solicite todos los nodos recibirá un valor **Vtam=1.0**
- **PriorityFavorSmall=NO** Si se fija este parámetro en **YES**, aquel trabajo que solicite un sólo **core** recibirá un valor **Vtam=1.0**
- **PriorityFlags=SMALL_RELATIVE_TO_TIME** El tamaño del trabajo en CPUs es dividido por el límite en minutos, y el resultado es dividido por el número total de CPUs.

$$Wtam = \frac{CPUsSolicitados}{TiempoSolicitado \times TotalCPUs} \tag{8.2}$$

- **PriorityWeightJobSize** Contribuye con el peso del factor de tamaño de trabajo.

El factor **FairShare** (punto flotante entre 0.0 y 1.0) refleja cuanto se comparte un recurso que un usuario ha localizado y la cantidad de recursos que el trabajo del usuario ha consumido.

- El componente **fair-share** influye en el orden en el cual un trabajo en cola es programado para ejecutarse.
- **PriorityWeightFairShare** el un entero sin signo que contribuye con este factor.

Se le puede asignar el factor **fairshare** a una categoría de contabilidad con el siguiente comando

```
sacctmgr add account general Description="General Accounting" \
Organization=SC3 fairshare=50
```

Para modificar este parámetro puede utilizar el siguiente comando

```
sacctmgr modify account where name=general cluster=guane \
set fairshare=30
```

Para visualizar los valores de este parámetro utilice el siguiente comando:

```
sacctmgr list assoc tree format=cluster,account,user.fairshare
```

Con el parámetro **PriorityUsageResetPeriod** se puede reiniciar el valor de fairshare. Los valores que toma son: NONE, NOW, DAILY, WEEKLY, MONTHLY, QUARTERLY y YEARLY

8.3. Programador Backfill

Backfill es el programador predefinido. Algunas de sus características son:

- Si no se utiliza el programador, cada partición es programada en orden estricto según la prioridad lo cual no es eficiente.
- Este permite iniciar los trabajos de baja prioridad que no tengan impacto en el tiempo de inicio de los trabajos de alta prioridad.
- Mientras mayor precisión se tenga en la especificación del tiempo del trabajo el programador backfill trabajará mejor (Educar a los usuarios).
- Backfill considera aspectos como la prioridad del trabajo, derecho de preferencia (preemption), programación conjunto (gang), recursos generales (gres), requisitos de memoria, etc.

La figura 8.1 ilustra el modo de operación de este programador.

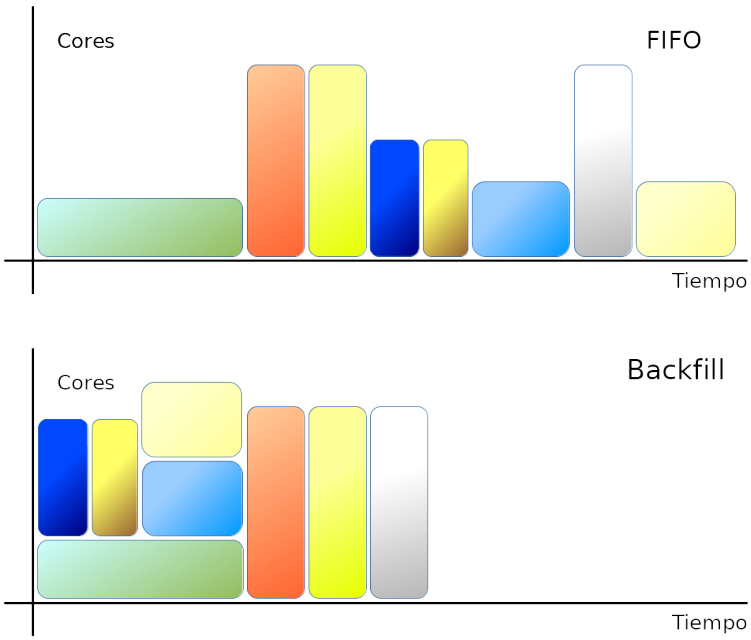

Figura 8.1: Programador Backfill

Los parámetros que sirven para configurar backfill son los siguientes:

- **DefaultTime** : Es el tiempo límite predefinido del trabajo. Se especifica en la partición.

- **MaxTime**: Es el tiempo máximo del trabajo y se especifica en la partición.
- **OverTimeLimit**: Es el tiempo en que un trabajo puede excederse una vez que haya alcanzado el tiempo máximo. Luego de este valor el trabajo es eliminado.

Debido a que backfill es una operación que consume tiempo, es necesario considerar algunos otros parámetros:

- **bf_continue**: Permite continuar a backfill despues de liberar los bloqueos (locks) para otras operaciones.
- **bf_interval**: Especifica el tiempo entre los intentos de backfill. El valor predefinido es 30 segundos.
- **bf_max_job_part**: Número máximo de trabajos a iniciarse por partición
- **bf_max_job_start**: Número máximo de trabajos a iniciarse en cada ciclo
- **bf_max_job_test**: Número máximo de trabajos a considerarse en la programación en cada ciclo. El valor predefinido es 100 trabajos.
- **bf_max_job_user**: Máximo número de trabajos a iniciarse por usuario en cada ciclo.
- **bf_resolution**: Tiempo de resolución de la programación de backfill. El valor predefinido es 60 segundos.
- **bf_window**: Tiempo para considerar trabajos futuros. El valor predefinido es un día.

8.4. Preemption

Esto se refiere a la preferencia de un trabajo para utilizar un recurso, es decir, su capacidad de utilizar los recursos que ya están siendo usados por otro trabajo [3]. Los siguientes parámetros controlan esta propiedad:

- **Preempt/none** No hay preferencia.
- **Preempt/QOS** Preferencia basada en la calidad de servicio.
- **Preempt/partition_prio** Preferencia basada en la prioridad.

Los mecanismos utilizados para utilizar el uso preferencial de los recursos son:

- **CANCEL** Elimina el trabajo.
- **SUSPEND** Suspende en memoria los trabajos de baja prioridad.

- **SUSPEND,GANG** Permite resumir el trabajo en cuanto los recursos están nuevamente disponibles. No es compatible con QoS.
- **CHECKPOINT** Realiza un checkpoint de los trabajos de baja prioridad.
- **REQUEUE** Coloca nuevamente en la cola el trabajo al que se le quitaron los recursos.
- **OFF** Ningún trabajo se suspende. No es compatible con QoS.

8.4.1. Propiedad GANG

El uso de **GANG** tiene sus ventajas y desventajas:

8.4.2. Ventajas

- Incrementa el rendimiento.
- Incrementa la utilización de los recursos.
- Reduce el tiempo de espera de un trabajo.

8.4.3. Desventajas

- Los trabajos largos tardarán más.
- Los trabajos permanecen en el nodo.
- Hay más impacto cuando hay fallas de hardware.

Para mostrar un ejemplo de configuración de la preferencia de uso de los recursos agregue las siguientes líneas al archivo **slurm.conf**

```
ProctrackType=proctrack/cgroup
SchedulerType=sched/gang
TaskPlugin=task/cgroup
SchedulerTimeSlice=60
PreemptType=preempt/partition_prio
PreemptMode=suspend,gang
PartitionName=DEFAULT Shared=FORCE:2 Nodes=knl[001-256]
PartitionName=low Priority=1 Default=YES
PartitionName=high Priority=2
```

Y agregue las siguientes líneas en el archivo **cgroup.conf**

```
CgroupAutomount=yes
CgroupReleaseAgentDir="/etc/slurm/cgroup"
AllowedDevicesFile="/etc/slurm/allowed_devices_file.conf"
```

8.4.4. Propiedad SUSPEND

Si se utiliza **SUSPEND** nos encontramos con una ventaja adicional que es la optimización de la carga del cluster, pero los trabajos largos tardan mucho más.

Para activar esta propiedad agregue las siguientes líneas en el archivo **slurm.conf**

```
ProctrackType=proctrack/cgroupTaskPlugin=task/cgroup
PreemptType=preempt/partition_prio
PreemptMode=suspend,gang
PartitionName=DEFAULT Shared=FORCE:1 Nodes=knl[001-256]
PartitionName=low Priority=1 Default=YES PreemptMode=SUSPEND
PartitionName=high Priority=2 PreemptMode=OFF
```

Y estas líneas en el archivo **cgroup.conf**

```
CgroupAutomount=yes
CgroupReleaseAgentDir="/etc/slurm/cgroup"
AllowedDevicesFile="/etc/slurm/allowed_devices_file.conf"
```

8.4.5. Propiedad CHECKPOINT

Si utilizamos el mecanismo de **checkpoint** se genera un checkpoint para los trabajos de menos prioridad y también se abre la posibilidad de migrar el trabajo. Sin embargo, esto introduce una recarga adicional al sistema, no siempre es adecuado y depende de la arquitectura. Para configurar esta propiedad agregue las siguientes líneas al archivo **slurm.conf**

```
CheckpointType=checkpoint/blcr
JobCheckpointDir=/BeeGFS/checkpointPreemptType=preempt/partition_prio
PreemptMode=CHECKPOINT
PartitionName=DEFAULT Shared=NO Nodes=knl[001-256]
PartitionName=low Priority=1 Default=YES PreemptMode=CHECKPOINT
PartitionName=high Priority=2 PreemptMode=OFF
```

Capítulo 9

Plugins

SLURM tiene muchas funcionalidades que se incorporar a través de **plugins**. Esta sección muestra algunos de los más utilizados.

9.1. Integración con MPI

SLURM se integra muy bien con varias implementaciones de MPI. En este ejemplo se describe como acoplar OpenMPI con SLURM

9.1.1. Descargar OpenMPI

```
sudo su -
cd /usr/local/src
wget https://www.open-mpi.org/software/ompi/v2.0/downloads/openmpi-2.0.0.tar.bz2
```

9.1.2. Compilar e Instalar OpenMPI

```
tar xvjf openmpi-2.0.0.tar.bz2
cd openmpi-2.0.0
export CFLAGS="-I/usr/local/slurm/include/slurm"

./configure --prefix=/usr/local/openmpi/ --enable-static \
--enable-mpi-fortran -enable-mpi-cxx --with-cuda \
--with-pmi=/usr/local/slurm/ --with-pmi-libdir=/usr/local/slurm/ \
CC=icc CC=icc CXX=icpc FC=ifort

make
make install
```

Las opciones de configuración –**with-pmi** y –**with-pmi-libdir** son las opciones que integran OpenMPI con SLURM

9.1.3. Variables de Ambiente

Configurar las variables de ambiente correspondientes creando un archivo en el directorio **/etc/profile.d** de todos los nodos del cluster con las siguientes líneas

```
#!/bin/sh
export PATH=/usr/local/openmpi/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openmpi/lib:$LD_LIBRARY_PATH
export MANPATH=/usr/local/openmpi/man:$MANPATH
```

9.2. Control de Acceso a los Nodos

En la mayoría de los casos es deseable que los usuarios no puedan ingresar a los nodos de cómputo para evitar que sus actividades provoquen un desbalanceo de la carga que SLURM trata de mantener. Para evitar esto, podemos intalar un plugin que integre SLURM con el sistema de autenticación de Linux **Pluggable Authentication Modules (PAM)**

Para integrar estos dos sitemas siga los siguientes pasos:

9.2.1. Compilar SLURM con Soporte PAM

```
./configure --prefix=/usr/local/slurm --enable-pam --with-pam_dir=/lib64/security
make
make contrib
make install
```

9.2.2. Configurar PAM en los Nodos de Cómputo

Copie la biblioteca compilada en el sistema operativo de todos los nodos de cómputo

```
cp contribs/pam/.libs/pam_slurm.* /lib64/security/
```

Luego, edite el archivo **/etc/pam.d/sshd** y modífiquelo para que se vea así:

```
#%PAM-1.0
auth       required     pam_sepermit.so
auth       include      password-auth
#####################################
account    required     pam_slurm.so
#####################################
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
#pam_selinux.so open should only be followed by sessions
#to be executed in the user context
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

Agregue la siguiente línea al archivo **slurm.conf** de todos los nodos y reinicie todos los demonios de SLURM

```
UsePAM=1
```

Finalmente, puede probar tratando de abrir una sesión ssh a uno de los nodos

```
ssh nodo1
```

Y ahora reserve un nodo e intente abrir la sesión de ssh al nodo asignado

```
salloc --nodes=1 --time=01:00:00
squeue
ssh nodoX
```

9.3. Checkpoint

SLURM está integrado con **Berkeley Lab Checkpoint/Restart (BLCR)** para proporcionar checkpoint y reinicio automático de trabajos. Dentro de las funcionalidades se tiene:

- Checkpoint completo de trabajos **batch** con todas sus tareas.
- Checkpoint periódicos
- Reinicio de la ejecución a partir de archivos generados desde un checkpoint
- Re encolamiento automático y reinicio de ejecución de trabajos batch

Cabe destacar que aún no puede gestionar trabajos interactivos

9.3.1. Instalar Berkeley Lab Checkpoint/Restart (BLCR)

La función de **checkpoint** de SLURM se apoya en BLCR. Para instalar este software siga las siguientes instrucciones

```
cd /usr/local/src
wget http://crd.lbl.gov/assets/Uploads/FTG/Projects/CheckpointRestart/downloads/blcr-0.8.5.tar.gz
tar xvzf blcr-0.8.5.tar.gz
cd blcr-0.8.5
mkdir builddir
cd builddir
../configure --prefix=/usr/local/blcr
make
make install
```

9.3.2. Recompilar SLURM

Para la integración es necesario indicar a SLURM en tiempo de compilación que incluya la biblioteca de BLCR

```
./configure --prefix=/usr/local/slurm --with-blcr=/usr/local/blcr --enable-pam \
--with-pam_dir=/lib64/security
make
make install
```

9.3.3. Configurar SLURM

Agregue la siguiente línea al archivo **slurm.conf** de todos los nodos y reinicie todos los demonios de SLURM

```
CheckpointType=checkpoint/blcr
```

9.3.4. Modo de uso

Se puede crear un checkpoint para un trabajo cualquier de dos formas:

- Ejecute el siguiente comando para crear un checkpoint desde fuera del contexto del trabajo

  ```
  scontrol checkpoint create <jobid>
  ```

- O puede configurar el propio trabajo para que genere cada cierto tiempo un checkpoint agregando las siguientes sentencias al script del trabajo.

```
#SBATCH --checkpoint <minutes>:<seconds>
#SBATCH --checkpoint-dir <directory>
```

  La primera de ellas indica la frecuencia de creación del checkpoint. La siguiente indica el directorio donde se crea el checkpoint. Es preferible que este directorio esté disponible en todos los nodos del cluster. Existen opciones correspondientes a estas sentencias en los comando srun y sbatch.

Capítulo 10

Recursos Generales (GRES)

SLURM es capaz de gestionar recursos genéricos distintos a los recursos tradicionales dentro de un cluster (Proceador, RAM, red, etc). Esto se hace especificando los nuevos recursos en un archivo texto ubicado en el directorio de la configuración de la instalación (**etc/gres.conf**).

10.1. Gestión de GPUs

Por ejemplo, si desea incorporar 4 **GPUs** puede ingresar las siguientes líneas dentro del archivo. Luego, reinicie los demonios de SLURM.

```
Name=gpu Type=Tesla File=/dev/nvidia0
Name=gpu Type=Tesla File=/dev/nvidia1
Name=gpu Type=Tesla File=/dev/nvidia2
Name=gpu Type=Tesla File=/dev/nvidia3
```

Para utilizar las GPUs agregue la siguiente opción a los comandos srun o sbatch

```
--gres=gpu:N
```

Donde **N** es la cantidad de GPUs que va a utilizar

10.2. Gestión de Disco

En este ejemplo se supone un par de directorios que se desean gestionar a través de SLURM. El primero ubicado localmente (/fast) y el segundo en un sistema de archivos distribuido (/data). Agregue las siguientes líneas al archivo **gres.conf**. Luego, reinicie los demonios de SLURM.

```
Name=disk Type=fast Count=48G
Name=disk Type=data Count=147G
```

Para solicitar el uso de estos recursos puede agregar las siguientes opciones a los comandos srun o sbatch

```
--gres=fast:30
```

Bibliografía

[1] Doublas Eadline. *High Performance Computing for Dummies*. Wiley Publishing, Inc, 2009.

[2] Inc Schmed. Multifactor Priority Plugin . https://slurm.schedmd.com/priority_multifactor.html, 2013.

[3] Inc Schmed. Preemption . https://slurm.schedmd.com/preempt.html, 2017.

[4] Inc Wikimedia Foundation. Slurm Workload Manager . https://en.wikipedia.org/wiki/Slurm_Workload_Manager, 2017.

[5] Andy B. Yoo, Morris A. Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *JSSPP*, volume 2862 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.