Universidad de Valladolid

E.T.S Ingeniería Informática

14 de Junio de 2016

Máster en Ingeniería Informática

Infraestructura para el Desarrollo de Aplicaciones de Computación de Altas Prestaciones

Instalación, configuración y uso de cluster con Rocks

Autores:
Diego González Serrador
José Martín Gago

Resumen

Rocks es una distribución de Linux para cluster de computadoras de alto rendimiento. Para instalar esta distribución se deberán configurar aspectos tales como la conectividad de red interna y externa al cluster y los paquetes a instalar para el nodo frontend. Posteriormente será necesario dar de alta los nodos encargados de realizar los trabajos del cluster. Rocks por defecto incluye el sistema de colas SGE y el servicio Ganglia, el primero será el encargado de repartir los trabajos entre los nodos del Cluster y con Ganglia podremos visualizar de manera gráfica el estado del cluster.

Ligado a Rocks, a la computación de alto rendimiento y a los grandes datacenters, la Universidad Politécnica de Valencia a desarrollado CLUES. CLUES es un sistema de gestión de energía para clusters de computación, cuya función principal es la de apagar los nodos internos del cluster cuando no están siendo utilizados y, de forma recíproca, encenderlos de nuevo cuando son necesarios.

Índice

Índice de figuras

1. Introducción y motivación

1.1. Estructura del documento

Este documento presenta la memoria del trabajo práctico realizado durante el curso 2015-2016 en la asignatura Infraestructura para el Desarrollo de Aplicaciones de Computación de Altas Prestaciones.

Las primeras secciones del documento describen y explican a grosso modo los conceptos de relativos a computación de alto rendimiento. Las siguientes partes se centran en enumerar las tareas realizadas para la realización de la práctica así como los problemas encontrados.Finalmente el trabajo concluye con una breve conclusión y mostrando las referencias citadas y consultadas.

1.2. Computación de alto rendimiento (HPC)

La computación de alto rendimiento (High Performance Computing)[4][6] consiste en la utilización de clusters, supercomputadores o mediante el uso de computación paralela. Actualmente, la mayoría de las ideas de la computación distribuida se basan en la computación de alto rendimiento.

La computación de alto rendimiento es aquella en la cual se proporciona una mayor capacidad de cómputo de la que se obtiene empleando máquinas individuales.

Un cluster, es un grupo de computadores independientes funcionando como uno solo. Se emplean para resolver problemas complejos. Estos clusters pueden dividirse en:

- HPC: alto rendimiento (aplicaciones altamente paralelas).
- HTC: alta productividad (aplicaciones con un gran número de ejecuciones).
- HAC: alta disponibilidad (tolerancia a fallos).
- LBC: balanceo de carga (gran número de accesos).

Podemos encontrar varias listas para establecer un ranking de ordenadores en función de distintas medidas:

- TOP 500: supercomputadoras con mayor rendimiento.
- Green 500: supercomputadoras con mayor rendimiento por eficiencia energética.
- Graph 500: supercomputadoras con mayor carga intensiva de datos.

2. Rocks cluster

Es una distribución de Linux para clusters de computadores de alto rendimiento. Rocks se basó inicialmente en la distribución de Red Hat, aunque actualmente están basadas en CentOS, con un instalador modificado que simplifica la instalación "en masa" en muchos computadores. Rocks incluye una gran cantidad de herramientas, las cuales no forman parte de CentOS pero son los componentes integrales los que convierten un grupo de ordenadores en un cluster.

2.1. Configuración de la BIOS

Antes de realizar la instalación del nodo frontal y de los nodos de computación, se revisó en todos ellos la configuración de la BIOS. Se comprobó que en todos los nodos que las opciones correspondientes a PXE[1][5] y WOL[2][8] estuvieran activadas.

2.2. Instalación del frontend

El primer paso para la instalación del cluster bajo el sistema operativo Rocks. Es instalar el sistema operativo en el computador que actuará como frontend. El frontend es el nodo maestro del sistema, desde él se instalarán los demás nodos vía PXE, será el encargado de enviar trabajos al resto de nodos, será el punto de acceso al cluster vía internet y además se encargará de apagar o encender más nodos en función de la carga de trabajo del cluster.

La instalación del frontend debe hacerse de manera manual. Los siguientes párrafos muestran el proceso de instalación seguido, así como las configuraciones realizadas.

[1] Preboot eXecution Environment (PXE) (Entorno de ejecución de prearranque), es un entorno para arrancar e instalar el sistema operativo en computadoras a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados.

[2] Wake on LAN (WOL, a veces WoL) es un estándar de redes de computadoras Ethernet que permite encender remotamente computadoras apagadas.

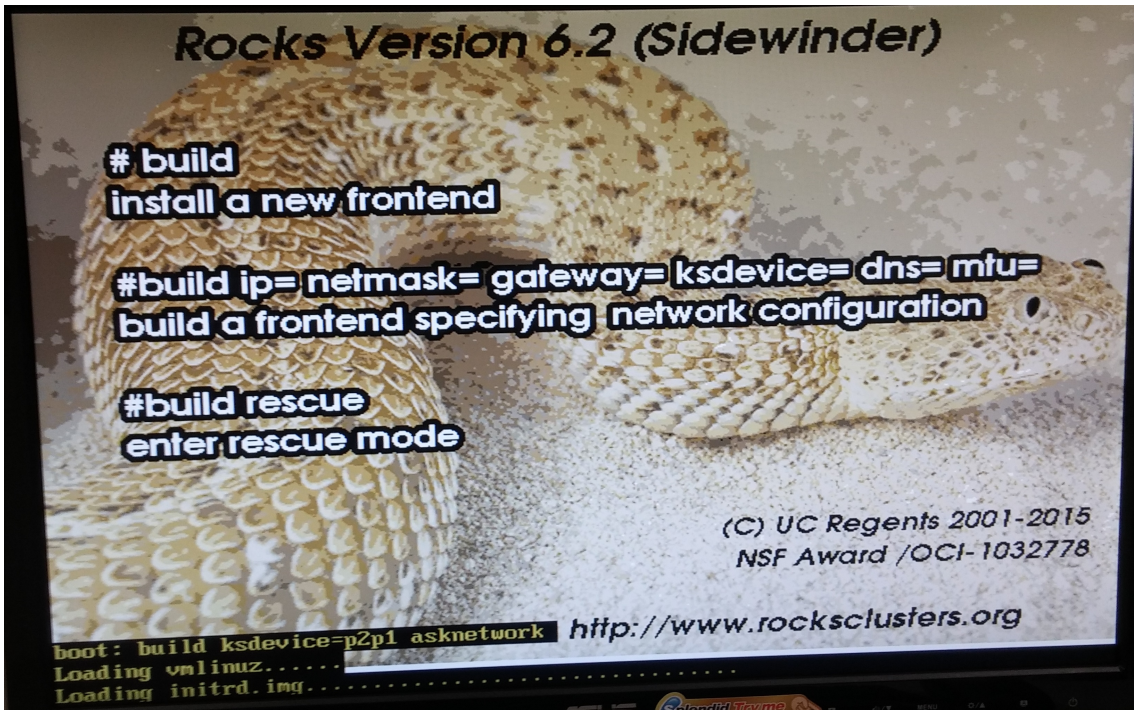

Figura 1: Portada instalación de Rocks

En las primeras etapas de la instalación debemos elegir el modo de configuración para la conexión de red. El frontend cuenta con dos interfaces de red, la primera de ellas será utilizada como punto de acceso al cluster desde Internet y por lo tanto contará con una ip pública. La segunda de estas interfaces de red proporcionará la comunicación con los nodos de computación del cluster, por su parte, los nodos de computación contaran únicamente con una interfaz de red para la comunicación interna. Las figuras 2 ,3 y 5 muestran la configuración de red elegida.

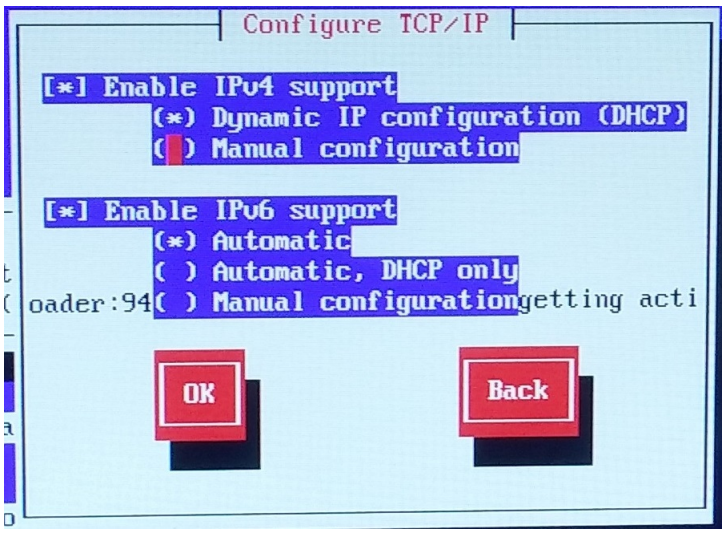

Figura 2: Configuración de la red en Rocks

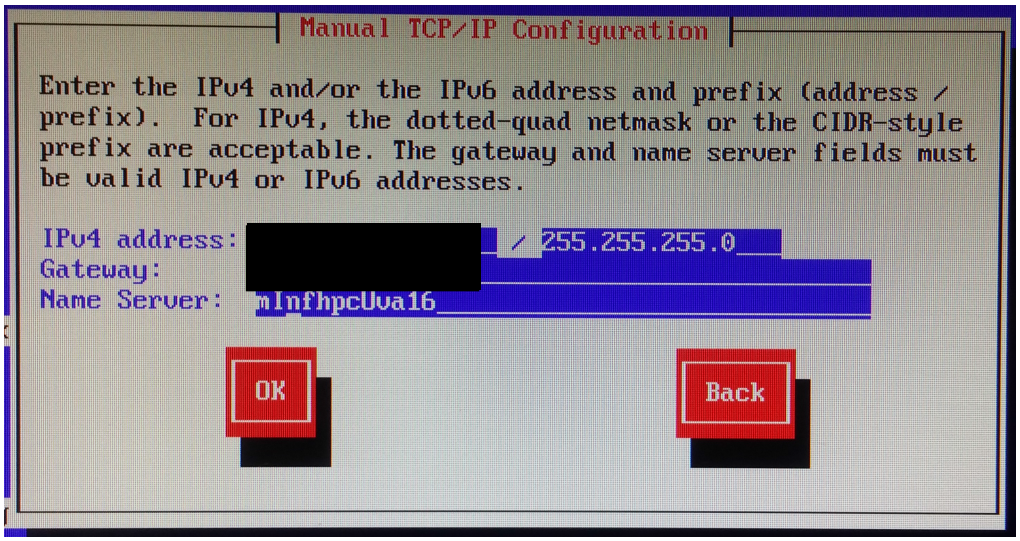

Figura 3: Configuración IP pública del frontal durante la instalación de Rocks

También, durante la instalación debemos seleccionar los paquetes que deseamos sean instalados junto al sistema operativo, en función de las necesidades del cluster elegiremos mas o menos componentes. La figura 4 muestra los paquetes elegidos.

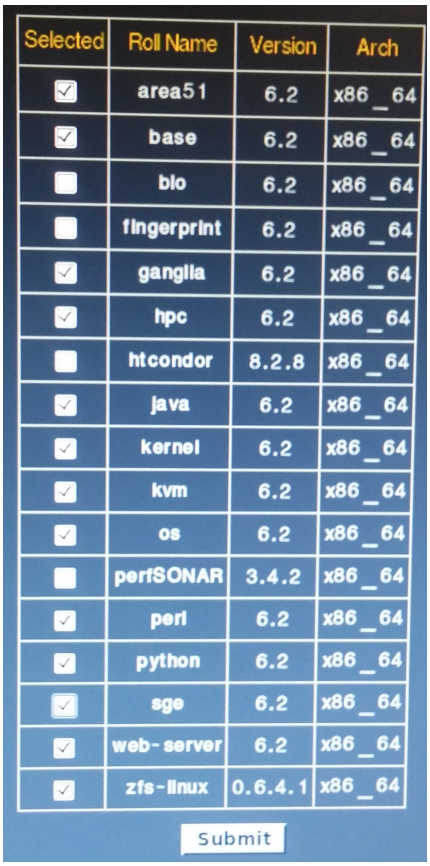

Figura 4: Selección de paquetes durante la instalación de Rocks

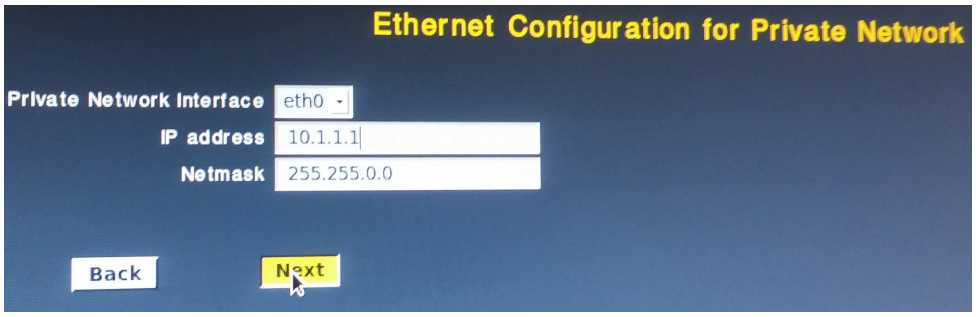

Figura 5: Configuración IP privada del frontal durante la instalación de Rocks

Finalmente y también como configuración de red debemos informar la puerta de acceso y añadir los servidores DNS para nuestro cluster. La figura 6 muestra los servicios seleccionados.

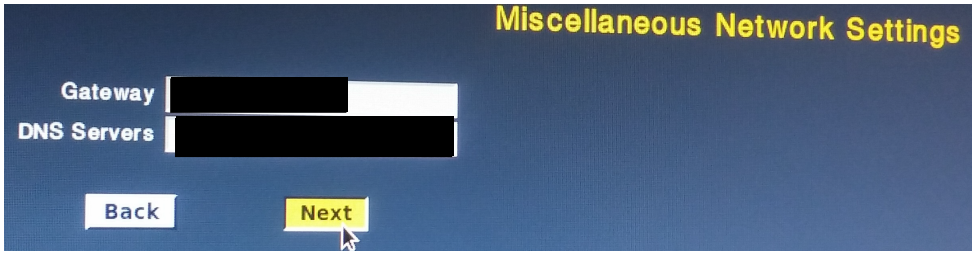

Figura 6: Configuración dns y gateway durante la instalación de Rocks

Rocks, además de la configuración de red automatizada que incluye durante la instalación, también permite configurar la red vía ficheros de configuración como cualquier otra distribución de Linux. Estos scripts en Centos 6 se pueden localizar en la ruta */etc/sysconfig/network-scripts/nombre_interfaz_red* y tienen el siguiente aspecto:

ifcfg-eth0:

```
DEVICE=eth0
TYPE=Bridge
IPADDR=10.1.1.1
NETMASK=255.255.0.0
BOOTPROTO="static"
ONBOOT=yes
MTU=1500
```

ifcfg-eth1:

```
DEVICE=eth1
TYPE=Bridge
IPADDR=[redacted]
NETMASK=255.255.255.0
GATEWAY=[redacted]
DNS1=[redacted]
DNS2=[redacted]
BOOTPROTO=static
ONBOOT=yes
```

```
10 MTU=1500
```

Los ejemplos anteriores muestran la configuración de red final elegida, en esta práctica, para el frontend. Si se realizase algún cambio más en estos ficheros de configuración sería necesario reiniciar el servicio de red:

```
1 hpcuva@rocks:/$ service network restart
```

2.3. Configuración del teclado

Al utilizar Rocks por primera vez nos encontraremos con el problema de que el teclado establecido es el inglés y por lo tanto tendremos dificultades a la hora de escribir comandos por el terminal, por ello lo más sencillo es cambiar el idioma establecido.

2.4. Instalación de nodos

Para iniciar la instalación de los nodos, introducimos el comando:

```
1 hpcuva@rocks:/$ insert-ethers
```

En la siguiente pantalla debemos seleccionar "Compute" tal y como muestra la figura 7. A continuación debemos ir arrancando el resto de los nodos de uno en uno y el programa los irá detectando e instalando, la figura 8 muestra esto. Sabremos que un nodo está instalado cuando se muestre con un asterisco, debemos esperar que esto ocurra antes de encender el siguiente nodo.

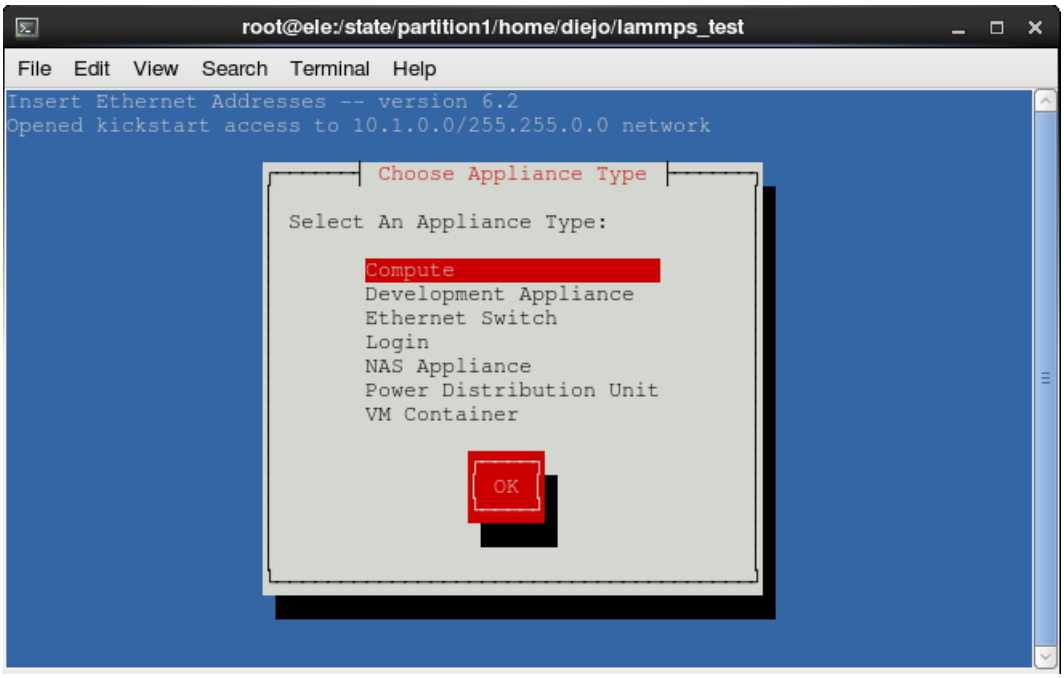

Figura 7: Comando insert-ethers

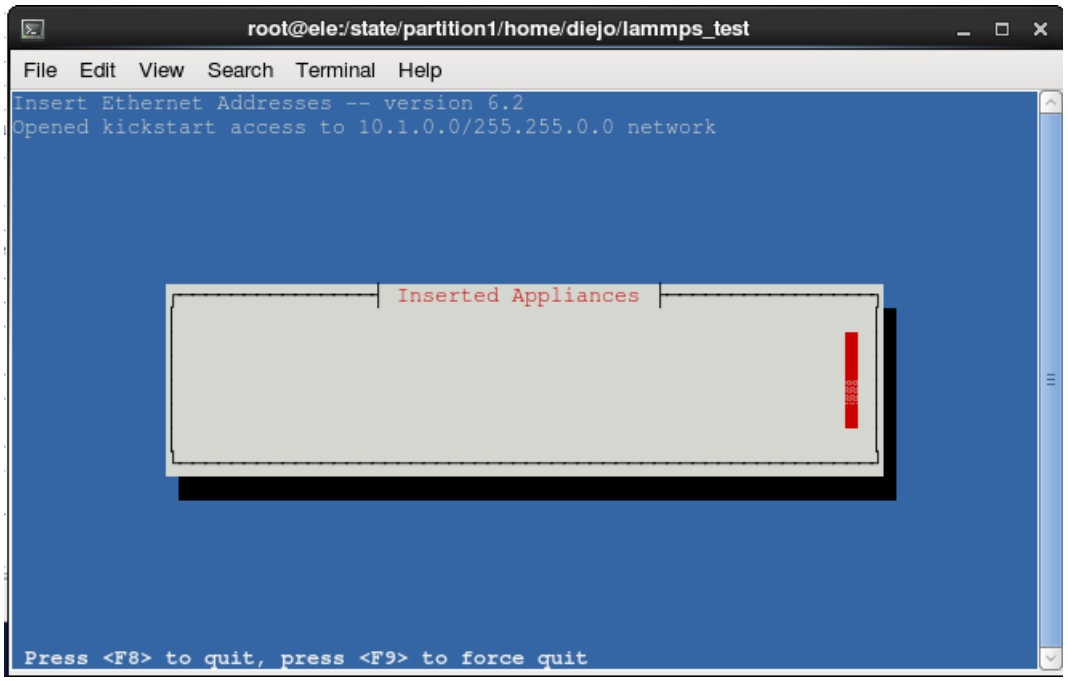

Figura 8: Comando insert-ethers, esperando nodos

Durante la instalación de los nodos en un principio tuvimos problemas debido a que los nodos no recibían la IP de la red interna y no podían comunicarse con el maestro.

Además de los problemas descritos, es importante saber que el primer nodo de computación (compute-0-0) tiene tendencia a tener errores y funcionar de manera anómala. Se ha llevado a cabo una re-instalación del nodo descrito en varias ocasiones mediante:

```
hpcuva@rocks:/$ rocks set host compute-0-0 action=install
```

Pero los problemas han continuado, por lo que probablemente se deba a algún componente hardware.

Finalmente podemos comprobar el estado del cluster Rock con los siguientes comandos:

```
[root@ele lammps_test]# rocks list host
HOST          MEMBERSHIP CPUS RACK RANK RUNACTION INSTALLACTION
ele:          Frontend    2    0    0    os        install
compute-0-0:  Compute     1    0    0    os        install
compute-0-1:  Compute     1    0    1    os        install
compute-0-2:  Compute     1    0    2    os        install
compute-0-3:  Compute     1    0    3    os        install
compute-0-4:  Compute     1    0    4    os        install
```

```
[root@ele lammps_test]# rocks list cluster
FRONTEND CLIENT NODES TYPE
ele:     ------------ physical
:        compute-0-0  physical
:        compute-0-1  physical
:        compute-0-2  physical
:        compute-0-3  physical
:        compute-0-4  physical
```

Y comprobar las conexiones de red de nuestro cluster con el siguiente comando:

```
[root@ele lammps_test]# rocks list host interface
HOST           SUBNET   IFACE MAC                IP              NETMASK
    MODULE NAME          VLAN OPTIONS CHANNEL
compute-0-0: private  eth0  [redacted]  10.1.255.254    255.255.0.0
    ------- compute-0-0 ----- -------- --------
compute-0-1: private  eth0  [redacted]  10.1.255.253    255.255.0.0
    ------- compute-0-1 ----- -------- --------
compute-0-2: private  eth0  [redacted]  10.1.255.252    255.255.0.0
    ------- compute-0-2 ----- -------- --------
compute-0-3: private  eth0  [redacted]  10.1.255.251    255.255.0.0
    ------- compute-0-3 ----- -------- --------
compute-0-4: private  eth0  [redacted]  10.1.255.250    255.255.0.0
    ------- compute-0-4 ----- -------- --------
ele:           private  eth0  [redacted]  10.1.1.1        255.255.0.0
    ------- ele          ----- -------- --------
ele:           public   eth1  [redacted]  [redacted]      255.255.255.0
    ------- ele          ----- -------- --------
```

2.5. Carga de trabajos, sistema de colas

Durante la instalación se ha instalado el sistema gestor de colas SGE. Podemos lanzar o encolar nuevos trabajos en el cluster con el comando:

```
hpcuva@rocks:/$ qsub runjob.sh
```

Podemos el estado de los trabajos así como el nodo en el que se está ejecutando:

```
[root@ele clues2]# qstat -f -u "*"
queuename                      qtype resv/used/tot. load_avg arch
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   0/1/1          0.00     linux-x64
     32 0.55500 test       root         r     06/17/2016 10:15:44     1
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   0/1/1          0.02     linux-x64
     34 0.55500 test       root         r     06/17/2016 10:15:44     1
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   0/1/1          0.00     linux-x64
     33 0.55500 test       root         r     06/17/2016 10:15:44     1
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   0/1/1          0.00     linux-x64
     35 0.55500 test       root         r     06/17/2016 10:15:44     1
---------------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   0/1/1          0.00     linux-x64
     36 0.55500 test       root         r     06/17/2016 10:15:44     1

############################################################################
 - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING J
############################################################################
     37 0.55500 test       root         qw    06/17/2016 10:15:42     1
     38 0.55500 test       root         qw    06/17/2016 10:15:43     1
     39 0.00000 test       root         qw    06/17/2016 10:15:44     1
     40 0.00000 test       root         qw    06/17/2016 10:15:44     1
```

Finalmente, si fuera necesario, podemos terminar un trabajo de forma manual de la siguiente forma:

```
hpcuva@rocks:/$ qdel <ID_TRABAJO>
```

2.6. Ganglia

Ganglia permite visualizar con una interfaz web (`http://localhost/ganglia`) el estado y la carga de trabajo de todos los nodos del cluster. La figura 9 (obtenida de [3]) muestra la apariencia de este servico.

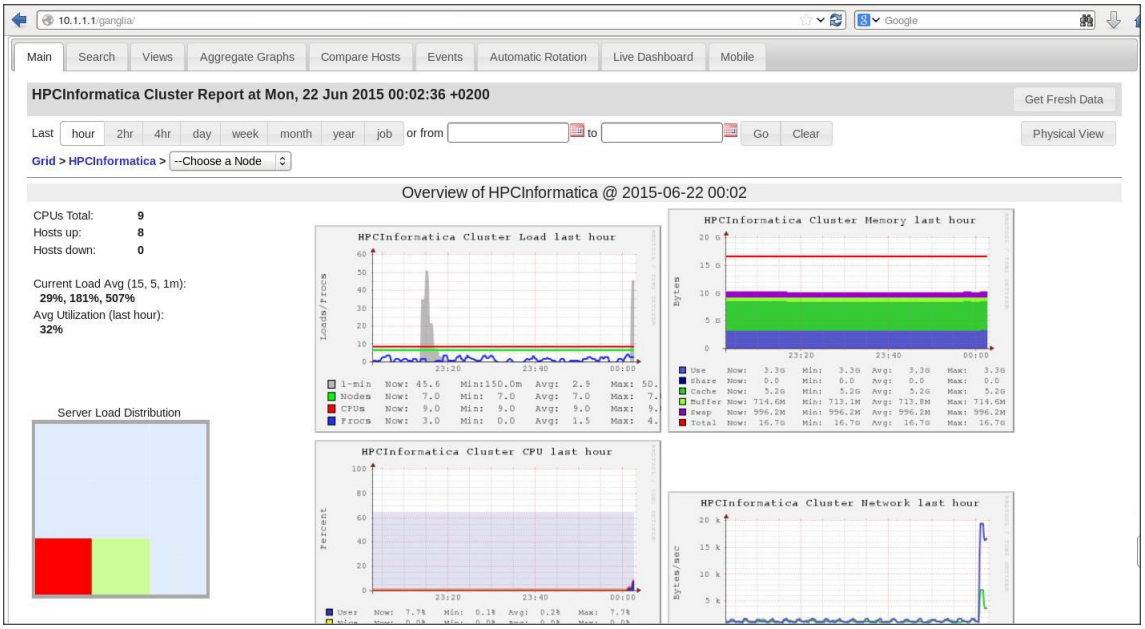

Figura 9: Interfaz web ofrecida por Ganglia

3. Clues

Clues es un sistema para la gestión de energía para clusters de altas prestaciones (HPC) e infraestructuras cloud, cuya función principal es apagar los nodos del cluster cuando no se están utilizando y encenderlos cuando son necesarios. Clues se integra con el middleware de gestión del cluster, como puede ser un gestor de colas o un sistema de gestión cloud.

Clues se integra con la infraestructura física utilizando plug-ins, para poder encenderlos y apagarlos mediante las técnicas más adecuadas para cada infraestructura.

La ventaja de Clues con respecto a otros sistemas de ahorro de enrgía, es que Clues se puede integrar con virtualmente cualquier gestor de recursos local. Clues puede integrarse con algunos de los sitemas de gestión de colas más populares y con dos de los sistemas de gestión de infraestructuras cloud privadas más conocidas (OpenNebula y OpenStack).

Desde la página de Clues (http://www.grycap.upv.es/clues/es/download.php) podemos acceder a su página de descarga (https://github.com/grycap/clues/) donde también están indicados los pasos para instalarlo.

3.1. Prerrequisitos

Para poder instalar Clues primero es necesario instalar Python Development Tools, Git y Clues Python Utils.

Para instalar Python Development Tools, buscamos el paquete necesario en función de si nuestro sistema operativo es de 32 o de 64 bits:

```
hpcuva@rocks:/$ yum search python | grep -i devel
```

Para instalar Git:

```
hpcuva@rocks:/$ yum install git
```

Para instalar Clues Python Utils:

```
hpcuva@rocks:/$ git clone https://github.com/grycap/cpyutils
hpcuva@rocks:/$ mv cpyutils /opt
hpcuva@rocks:/$ cd /opt/cpyutils
hpcuva@rocks:/$ python setup.py install --record installed-files.txt
```

3.2. Instalación de Clues

Tras ello podemos descargar Clues desde github https://github.com/grycap/clues/ siguiendo las indicaciones de la página.

```
hpcuva@rocks:/$ git clone https://github.com/grycap/clues
hpcuva@rocks:/$ mv clues /opt
hpcuva@rocks:/$ cd /opt/clues
hpcuva@rocks:/$ python setup.py install --record installed-files.txt
```

Ahora debemos configurarlo, para ello editaremos el fichero clues2.cfg:

```
hpcuva@rocks:/$ cd /etc/clues2
hpcuva@rocks:/$ cp clues2.cfg-example clues2.cfg
```

En LRMS_CLASS seleccionamos el plugin sge y en POWERMANAGER_CLASS seleccionamos wol.

```
# The python module that CLUES will use to monitorize the LRMS.
It MUST have a "lrms" class inside and it MUST be accesible by
python as an import
# * Tip: if neeeded, you can modify the cluesd executable to
include the path in which the module is using
sys.path.append("/path/to/module")
LRMS_CLASS=cluesplugins.sge

# The python module that CLUES will use to power on or off the
internal nodes. It MUST have a "powermanager" class inside and it
MUST be accesible by python as an import
# * Tip: if neeeded, you can modify the cluesd executable to
include the path in which the module is using
sys.path.append("/path/to/module")
POWERMANAGER_CLASS=cluesplugins.wol
```

Figura 10: Configuración de gestor de colas y de energía

Establecemos sge como el módulo de python que Clues utilizará para gestionar los recursos locales y wol será el módulo de python que se encargará de encender y apagar las máquinas de los nodos y de esta gestionar el gasto de energía.

Establecemos la lista de clases python que se usaran como planificadores:

```
# Comma separated list for the python classes that are used as
schedulers. The scheduling pipeline will be called in the same
order that are stated here. Each class MUST have a "schedule"
method inside and the class MUST be accesible by python as an
import
# * Tip: if neeeded, you can modify the cluesd executable to
include the path in which the module is using
sys.path.append("/path/to/module")
SCHEDULER_CLASSES=clueslib.schedulers.CLUES_Scheduler_Reconsider_
Jobs, clueslib.schedulers.CLUES_Scheduler_PowOff_IDLE,
clueslib.schedulers.CLUES_Scheduler_PowOn_Free
```

Figura 11: Establecimiento y orden de los planificadores

Estos planificadores se encargarán de planificar los trabajos, apagar y encender los nodos.

Editamos el fichero de configuración de wol /etc/clues2/conf.d/plugin-wol.cfg cambiando:

- El comando de encendido:

```
hpcuva@rocks:/$ ether-wake -i eth0 MAC
```

- El comando de apagado:

```
hpcuva@rocks:/$ ssh NOMBRE-NODO 'shutdown -h now'
```

Como puede verse en la imagen siguiente:

```
# * you can use %%a to substitute the MAC address and %%h to
substitute the hostname
WOL_CMDLINE_POWON=/sbin/ether-wake -i eth0 %%a
WOL_CMDLINE_POWOFF=ssh %%h 'shutdown -h now'
```

Figura 12: Configuración de wol

Además, es necesario añadir el fichero que contenga el nombre de los nodos y su mac, fichero que debemos crear y que en nuestro caso hemos llamado wol.hosts, como vemos en la siguiente imagen.

```
[WOL]
# The file in /etc/hosts format-like that contains the
correspondence of the MAC addresses to the host names in the LRMS
WOL_HOSTS_FILE=wol.hosts
```

Figura 13: Configuración de wol

Finalmente, para poder acceder a los nodos utilizando su nombre de nodos editamos el fichero /etc/ethers añadiendo el nombre de los nodos y su mac. El fichero wol.hosts tiene la misma estructura interna.

```
compute-0-0
compute-0-1
compute-0-2
compute-0-3
compute-0-4
```

Figura 14: Estructura del fichero

Para generar el fichero wol.hosts y editar el fichero /etc/ethers utilizamos el script *B. Script wake on lan* que se puede encontrar en los anexos. Este script obtiene la mac de los nodos utilizando para ello rocks.

4. Wake on Lan (WOL)

Wake on Lan es un estándar de redes de computadoras Ethernet que permite encender de manera remota computadoras apagadas. WOL debe estar activado en la configuración de la BIOS de la placa base. Puede ser necesario configurar el ordenador para reservar energía para la tarjeta de red cuando el ordenador esté apagado y también puede ser necesario activarlo desde la configuración de la tarjeta de red.

Además de la configuración de la BIOS, es necesario configurar los nodos para poder ser encendidos a través de la red. Para ello, editamos el fichero /etc/sysconfig/network-scripts/ifcfg-eth0 añadiendo al final:

```
ETHTOOLS_OPTS=''wol g''
```

Y ejecutar:

```
hpcuva@rocks:/$ ethtools -s eth0 wol g
```

Esta tarea se puede automatizar mediante el script *A. Script edición interfaces* que se puede encontrar en los anexos. Este script se encarga de recorrer los nodos y mediante una conexión ssh edita el fichero de red del nodo añadiendo una línea y ejecuta el comando para permitir el Wake on Lan.

5. Optimización de nodos

Uno de los objetivos de este trabajo era optimizar el consumo energético, algo muy importante en grandes datacenters, mediante el apagado y encendido de los nodos de computación en función de la carga de trabajo del cluster. Utilizando clues y unos pequeños scripts creados para esta práctica se consiguió esta meta, como anexos, se adjuntan 2 archivos(ambos debidamente documentados para entender su funcionamiento) para controlar si los nodos están o no activos y si están realizando tareas:

- ScriptNodos.sh: Se encarga de apagar y encender los nodos para ahorrar energía.
- Nodos2.sh: Se encarga de encender los nodos a los cuales se han asignado trabajo cuando estaban apagados, esto es debido a que el nodo maestro tiene retardos para detectar cuando el nodo se apaga y puede asignarle un trabajo sin saber que en realidad está apagado.

Los scripts comentados deberían ejecutarse de manera constante en el nodo *frontend* del cluster. El primero de los scripts es, básicamente, un proceso que se ejecuta de manera continuada una vez iniciado, debido a que el primer nivel es un bucle infinito. Esto implica que una vez ejecutado este proceso, la gestión automática de los nodos apagados o encendidos se realizará de manera constante siempre y cuando este proceso no sea finalizado. El segundo script es llamado de manera automática por el primer script, de forma que no implica ninguna configuración o ejecución a mayores, es ejecutado de manera automática después de un tiempo configurable que ha sido marcado en 9 minutos[3]. Como trabajo futuro sería aconsejable configurar la ejecución de este proceso como un servicio Linux que, además, fuera iniciado automáticamente al arrancar el sistema, de esta manera el administrador podría no ocuparse de la gestión del gasto energético.

El código fuente de estos dos scripts puede ser consultado en los anexos de este mismo informe; sin embargo a continuación se añaden unos diagramas UML de actividad que ayudarán al lector a comprender el funcionamiento de estas rutinas.

[3]Este tiempo ha sido obtenido de manera empírica en las pruebas de laboratorio para el entorno montado; pero puede ser recomendable ajustarlo en otros clusters.

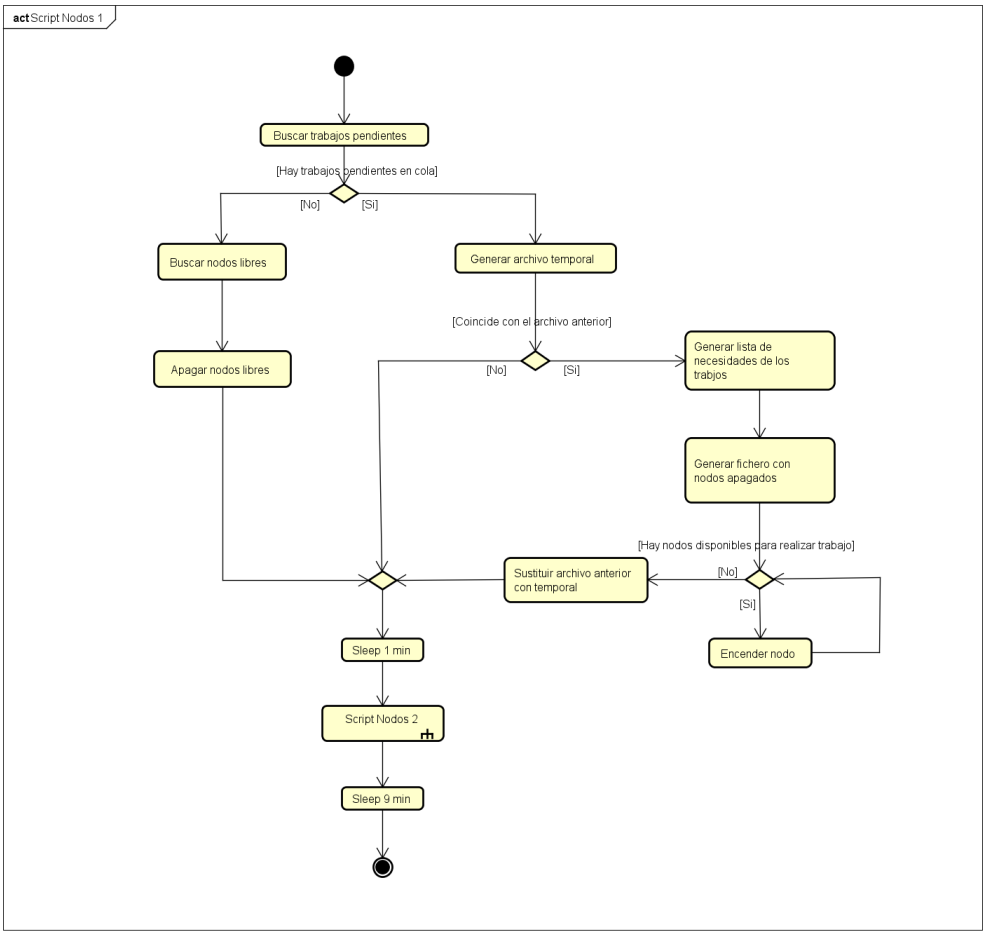

Figura 15: Diagrama de actividad del proceso seguido para encender o apagar nodos

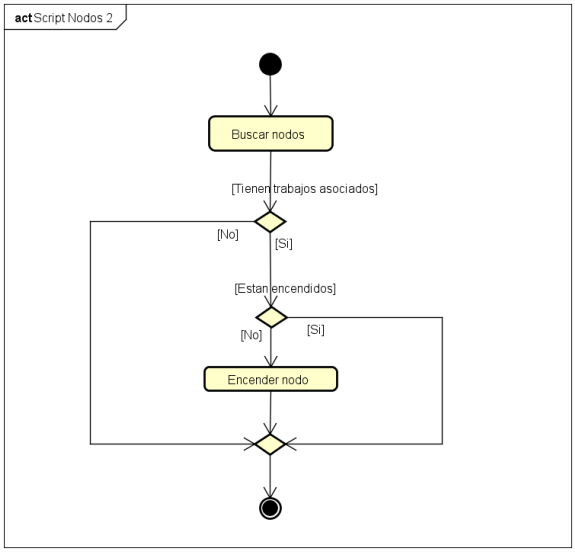

Figura 16: Diagrama de actividad del script auxiliar utilizado para evitar que los trabajos se queden en nodos apagados

Referencias

[1] Santos Tejido.I. *Infraestructuras para el desarrollo de aplicaciones de computación de altas prestaciones, Tema 1 - Estrategias de diseño de clusters*, (2016) .

[2] Barroso García.V y Rodriguez Gutiez.E. Universidad de Valladolid. *Instalación, configuración y puesta en producción de un cluster Rocks.*(2014). Consultado por última vez 08/06/2016, en: `http://www.ele.uva.es/~ivasan/HPC/abstracts/2013_2014_Veronica_Eduardo.html`

[3] Cerezo Redondo.B. Universidad de Valladolid. *Instalación, configuración y manejo de un cluster Rocks.*(2015). Consultado por última vez 08/06/2016, en: `http://www.ele.uva.es/~ivasan/HPC/abstracts/2014_2015_Borja.html`

[4] Wikipedia. *Computación de alto rendimiento.*(2016). Consultado por última vez 05/06/2016, en: `https://es.wikipedia.org/wiki/Computaci%C3%B3n_de_alto_rendimiento`

[5] Wikipedia. *Preboot Execution Environment.*(2016). Consultado por última vez 05/06/2016, en: `https://es.wikipedia.org/wiki/Preboot_Execution_Environment`

[6] Wikipedia. *Supercomputer HPC.*(2016). Consultado por última vez 05/06/2016, en: `https://en.wikipedia.org/wiki/Supercomputer`

[7] Wikipedia. *Rocks Clusters.*(2016). Consultado por última vez 05/06/2016, en: `https://es.wikipedia.org/wiki/Rocks_Clusters`

[8] Wikipedia. *Wake on Lan.*(2016). Consultado por última vez 05/06/2016, en: `https://es.wikipedia.org/wiki/Wake_on_LAN`

[9] Universidad Politécnica de Valencia. *Clues - cluster energy saving (for hpc and cloud computing).*(2016). Consultado por última vez 05/06/2016, en: `http://www.grycap.upv.es/clues/es/index.php`

[10] Universidad Politécnica de Valencia. *Clues source code in Github.*(2016). Consultado por última vez 05/06/2016, en: `https://github.com/grycap/clues/`

A. Script edición interfaces

```
#/bin/bash

# Numero total de nodos
num_nodos=`cat /etc/ethers | tail -n +3 | wc -l`

# Se recorren los nodos
for cont in $(seq 0 $((num_nodos-1)) )
do


  # Se anyade la linea al fichero de la interfaz
  ssh compute-0-$cont `echo ETHTOOL_OPTS=\''wol g\'' >> /etc/sysconfig/
    networkscripts/ifcfg-eth0`

  # Se ejecuta el comando
  ssh compute-0-$cont `echo ethtool -s eth0 wol g`


done
```

B. Script wake on lan

```
#!/bin/bash
for cont in $(seq 0 4)
do

  # Se obtiene la mac
  mac=$(rocks run host compute-0-$cont ifconfig | grep HWaddr | cut -f 3 -d
    "r")

  # Se guarda la mac y el nombre del nodo en los dos ficheros
  echo $mac compute-0-$cont >> wol.hosts
  echo $mac compute-0-$cont >> /etc/ethers
done
```

C. Script Nodos 1

```
#/bin/bash

# Definicion de tiempo de espera, para lanzar el segundo script y para
    volver a ejecutar el bucle infinito
tiempo_script=1m
tiempo_vuelta=9m

# Numero total de nodos
num_nodos=`cat /etc/ethers | tail -n +3 | wc -l`

# Bucle infitino
while :
do

  # Total de lineas en cola
  en_cola=`qstat -u "*" -s p | wc -l`

  # Total en cola con error
  cola_err=`qstat -u "*" -s p | grep "Eqw" | wc -l`

  # Numero de trabajos en cola validos
  cola_valid=`expr $en_cola - $cola_err`

  # Lista de trabajos pendiente Vacia
  if [ $cola_valid -eq 0 ]; then
#   echo "Vacia"

    # Se recorren los nodos
    for cont in $(seq 0 $((num_nodos-1)) )
    do

      # Se comprueba si estan libres
      if [ `qstat -f -u "*" -q all.q@compute-0-$cont.local | wc -l` -eq 3
    ]; then
#       echo $cont " Libre"

        # Como el nodo esta libre lo apagamos
        ssh compute-0-$cont 'shutdown -h now'

      fi

    done

  else
#   echo "No vacia"

    # Generamos el archivo temporal de Pendings
    qstat -u "*" -s p | tail -n +3 > pendingAux.txt

    # Ha cambiado la lista de espera (o es la primera vez que se lanza)
    if [ `diff pending.txt pendingAux.txt | wc -l` -ne 0 ]; then

      # Inicializar lista de nodos apagados
```

```
52      echo "" > Apagados.txt
53      num_Apagados=0
54
55      # lista con las necesidades para los trabajos
56      lst_Necesita=[]
57      lst_Necesita=`cat pendingAux.txt | rev | cut -d " " -f 9`
58
59
60      # Buscamos los nodos apagados
61      # Se recorren los nodos
62      for cont in $(seq 0 $((num_nodos-1)) )
63      do
64        # Si no es vacio ha habido error. Nodo apagado
65        if [ `ping -c 1 -q compute-0-$cont | grep "1 errors" | wc -l` -ne 0
    ]; then
66
67          # Anyadimos el nodo apagado al fichero
68          echo "compute-0-$cont" >> Apagados.txt
69
70          # Incrementamos el numero de apagados
71          num_Apagados=`expr $num_Apagados + 1`
72
73        fi # cierre if ping
74
75      done # Cierre for cont
76
77#     echo "Buscando apagados fin"
78
79      cont=1
80
81      # Se recorre la lista de necesidades de los trabajos
82      for item in $lst_Necesita;
83      do
84#       echo "Levantando para $item"
85        # Se comprueba si hay suficientes nodos apagados para servir el
    trabajo
86        if [ $item -le $num_Apagados ]; then
87
88
89          cont2=0
90          while [ $cont2 -lt $item ]
91          do
92
93
94            # Lista con los nodos apagados
95            lst_Apagados=[]
96            lst_Apagados=`cat Apagados.txt | tail -n +$cont`
97
98            # Se levanta el nodo
99            for encender in $lst_Apagados;
100           do
101#             echo "ether-wake -i eth0 $encender"
102             ether-wake -i eth0 $encender
103             break
104           done
105
```

```
106             # Se incrementan las variables de control
107             ((cont2+=1))
108             ((cont+=1))
109             ((num_Apagados-=1))
110           done
111         fi
112       done
113
114       # Se guarda en el archivo pending
115       cat pendingAux.txt > pending.txt
116     fi
117
118   fi
119
120   # Se espera un tiempo para lanzar el segundo escript
121   sleep $tiempo_script
122   sh ./Nodos2.sh
123
124   # Se espera hasta volver a realizar el bucle
125   sleep $tiempo_vuelta
126
127
128 done # Bucle infinito
```

D. Script Nodos 2

```
#/bin/bash

# Numero total de nodos
num_nodos=`cat /etc/ethers | tail -n +3 | wc -l`

# Se recorren los nodos
for cont in $(seq 0 $((num_nodos-1)) )
do

  # Se comprueba que no estan libres
  if [ `qstat -f -u "*" -q all.q@compute-0-$cont.local | wc -l` -ne 3 ];
   then


    # Si no es vacio ha habido error. Nodo apagado
    if [ `ping -c 1 -q compute-0-$cont | grep "1 errors" | wc -l` -ne 0 ];
   then

      # Encendemos el nodo
      ether-wake -i eth0 compute-0-$cont


    fi

  fi

done
```