LINUX CLUSTERING CON ROCKS
UNA GUÍA PRÁCTICA

Jorge Zuluaga, Dr.
Grupo de Física y Astrofísica Computacional, FACom
Instituto de Física
Universidad de Antioquia

Universidad de Antioquia
Medellín – Colombia
2006

Presentación

Se presenta en este documento una Guía Práctica al uso y administración básica de Clusters Beowulf que corren Linux Rocks como distribución de base. Por "práctica" entendemos aquí una guía capaz de introducir de forma expedita al usuario de un cluster que corre Rocks, en la utilización de algunas de las herramientas con las que cuenta la plataforma para su monitoreo, control y uso para el cálculo en Ciencias e Ingeniería. Se ofrecen también los elementos básicos de la administración cotidiana de un Cluster con Rocks.

El usuario de esta Guía deberá estar familiarizado con el uso del sistema operativo Linux. Aunque no se requieren conocimientos avanzados de uso y administración del sistema operativo, se asumirá que se cuenta con una experiencia básica en el manejo de la línea de comando, manipulación de archivos y directorios y tareas básicas de comunicación.

La Guía se estructura en tres partes. Una guía práctica del usuario, donde se asume que se dispone ya de una plataforma operativa, y que el lector busca solución inmediata al problema de la conectividad en el cluster, el uso y organización de los archivos y el lanzamiento de trabajos de computo usando las herramientas de HPC de las que dispone. Una guía práctica de administración que presenta soluciones rápidas a los problemas básicos de la administración del cluster, administración de sistemas de archivos, configuración de servicios y software, administración de usuarios, instalación de nuevo software y monitoreo del cluster. Y un conjunto de apéndices donde se exponen soluciones concretas a problemas específicos en el cluster, instalación de la plataforma, instalación de paquetes y librerías específicas, configuración personalizada de algunos servicios, entre muchas otras. Esta última parte es la más flexible del documento en tanto crecerá conforme surja la necesidad de abordar esos mismos temas.

La guía que presentamos aquí no pretende ser ni convertirse en un "manual de referencia" completo de la distribución ni de las herramientas que se usan en ella para hacer HPC y Grid Computing. Para ello están las referencias bibliográficas que se incluyen al final de la guía. Su estructura y contenido han sido diseñados de modo que puedan crecer conforme las necesidades expresadas por los usuarios de este tipo de sistemas en el contexto en el que la guía busca tener su principal audiencia (grupos de investigación científica o en ingeniería que usen la herramienta para investigación) así lo requieran.

La guía es una compilación elaborada para un curso taller que en el tema ofrecí al Grupo de Investigación en Química de Recursos Energéticos y Medio Ambiente en la Universidad de Antioquia. Aprovecho para agradecer al Grupo QUIREMA y en especial al Profesor Fanor Mondragón, que dispuso de los recursos económicos y físicos para la realización del mencionado curso. En la guía se recoge la experiencia que en los últimos años he acumulado en el montaje, uso y administración de Clusters Beowulf usando la distribución Linux Rocks. En ese sentido agradezco al Instituto de Física que me ha proveído de los recursos físicos y la descarga académica para dedicar parte de mi tiempo en la adquisición de tan valiosa experiencia.

Esta guía viene siendo activamente utilizada en la realización de cursos y talleres de capacitación en el uso de clusters con Rocks en la Universidad de Antioquia, donde se encuentran ya operativos 3 clusters dedicados a la investigación en Física y Química. El

material contenido aquí viene siendo también utilizado en cursos y talleres en los pregrados y posgrados de Física, Química e Ingeniería que usan la plataforma (cursos de Física y Astrofísica Computacional, curso de Programación sobre plataformas de computación distribuida, seminario de procesamiento paralelo, taller de computación científica). Se espera además que la guía pueda ser utilizada por la comunidad creciente de usuarios de Rocks en el país en particular en el marco de la iniciativa de conformación de un Grid Nacional a partir de la reunión de clusters y grids institucionales.

Actualizaciones de la guía pueden ser adquiridas a través del sitio del Grupo de Física y Astrofísica Computacional (FACom) http://urania.udea.edu.co/facom o del sitio del portal en Internet del Cluster del Instituto de Física, http://hercules.udea.edu.co

Jorge Zuluaga, Medellín, 2006

Convenciones

La guía contiene una colección de comando útiles, tips de uso, scripts y comentarios generales sobre un conjunto amplio de tópicos relacionados con el uso y administración de Clusters Rocks.
Se asume en esta guía las siguientes convenciones tipográficas:

Convención	**Explicación**	**Ejemplo**
Comandos #,#: # comando	*Comandos de linux. Todos los comando vienen numerados de acuerdo a la sesión en la que aparecen. El nombre del comando se indica seguido del símbolo del sistema: '#' cuando el comando debe ser ejecutado como administrador y '$' cuando el comando debe ser ejecutado como usuario.*	Comando 1,20: # ls /var/log/message*

Salida:
`salida en pantalla`
	Cuando se muestre la salida de un comando esta se presenta en tipo `courier` pequeño.	Salida:
		`/var/log/messages`
		`/var/log/messages.2`
		`/var/log/messages.4`
		`/var/log/messages.1`
		`/var/log/messages.3`

Archivo:
`...`
`#Configuration file`
`FILE=configuration`
`SERVER=localhost`
`...`
	El contenido de un archivo se presenta en tipo courier con una sangría respecto al resto del documento y con una línea vertical que delimita el contenido del archivo.
	Cuando el documento no esta completo se colocan puntos suspensivos arriba o abajo indicando la presencia de más líneas
		./run.sh:
		`#!/bin/bash`
		`#Script para SGE`
		`#-S /bin/bash`
		`#-j y`
		`dir=/home/fulano/run1`
		`cd $dir`
		`./program.out`

Descargue: archivo	*Cuando en una situación dada se requiera un archivo o un paquete especial se referirá al lector al sitio del Grupo FACom donde podrá encontrar el archivo respectivo.* *El sitio es:* *http://urania.udea.edu.co/facom* *Y los archivos se encuentran en el enlace "documentación"*	Descargue: run.sh

Primera Parte
Guía Práctica del Usuario

Guía 1
Guía práctica a la plataforma

Contenido sintético

1.Guía práctica a la plataforma

1.1.Acceso al frontend y a los nodos del cluster

Todo Cluster Linux configurado con Rocks tiene 2 tipos de máquinas: el frontend (o servidor) donde se centraliza la información sobre la plataforma, se crean las cuentas de usuario y donde se ejecutan los servicios principales del Cluster; y los nodos de computo (o simplemente los nodos) que son las maquinas en el cluster que realizan efectivamente los trabajos de computo en la plataforma o que se pueden utilizar para almacenar grandes volúmenes de información de manera distribuida.

El acceso de un usuario al cluster comienza con la apertura de una cuenta en el frontend a cargo del servidor del cluster. Para ello el usuario debe proveer un nombre de usuario válido (login) y asignar una contraseña también valida o aceptable para autenticarse en el frontend.

Una vez se tiene una cuenta creada es importante consultar al administrador el Fully Qualified Host Name (FQHN) del frontend, que es el nombre y el dominio con el que se reconoce la maquina en la red de la institución a la que pertenece o en Internet, y el IP (Internet Protocol address) que es la identificación numérica única de la maquina en Internet. Ambos el FQHN y el IP pueden ser solo reconocidos dentro de redes locales (una red institucional por ejemplo). Es importante consultar también con el root el dominio desde el que se puede acceder al frontend.

En los ejemplos sucesivos asumiremos para el FQHN de nuestro cluster cluster.dominio y para su IP 192.168.3.2.

El acceso al frontend en primera instancia se hace solicitando una terminal remota (shell remoto) a través del protocolo SSH (secure shell). La terminal puede abrirse usando lo que se conoce como un cliente de terminal remota ssh.

Acceso desde Linux

La mayoría de las máquinas Linux vienen dotadas de un cliente ssh que se invoca directamente desde el símbolo del sistema con el comando ssh.

Comandos 1,1:

$ ssh fulano@cluster.dominio
$ ssh fulano@192.168.3.2

```
Last login: Tue Oct 31 09:12:36 2006 from 192.168.3.2
Rocks 4.2.1 (Cydonia)
Profile built 00:06 13-Oct-2006

Kickstarted 19:36 12-Oct-2006
Rocks Frontend Node - Cluster

It doesn't appear that you have set up your ssh key.
This process will make the files:
     /home/fulano/.ssh/id_rsa.pub
     /home/fulano/.ssh/id_rsa
     /home/fulano/.ssh/authorized_keys

Generating public/private rsa key pair.
Enter file in which to save the key (/home/fulano/.ssh/id_rsa):
Created directory '/home/fulano/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/fulano/.ssh/id_rsa.
Your public key has been saved in /home/fulano/.ssh/id_rsa.pub.
The key fingerprint is:
30:8c:99:71:db:38:a2:91:99:9e:19:5d:ca:f7:01:c7 fulano@cluster.dominio
```

Una vez el usuario se logea por primera vez el frontend automáticamente configura el sistema de autenticación automática rsa del cliente ssh en el cluster. Este sistema es extremadamente importante porque garantiza que el usuario pueda acceder transparentemente a todos los recursos del cluster sin necesidad de proveer en todas las acciones su contraseña de usuario. La configuración del sistema de autenticación automático requiere proveer la que se conoce como una “passphrase”. Se recomienda dejar en blanco este parámetro. La configuración del rsa ocurre solamente la primera vez que el usuario se logea en el frontend.

Tips útiles

El comando ssh tiene una serie de opciones que permiten ajustar su comportamiento y determinar algunas propiedades importantes de la terminal remota. Así por ejemplo

Comandos 1,2:

$ ssh -XY fulano@cluster.dominio

Permitirá que desde la terminal remota que se este abriendo puedan ejecutarse aplicaciones gráficas que corran directamente sobre el frontend. Las opciones X y Y habilitan lo que se conoce en el mundo de ssh como X forwarding. Este mecanismo sin embargo hace un poco más lenta la comunicación, que puede finalmente impactar el rendimiento de los programas que se ejecuten sobre el cluster a posteriori.

En la mayoría de los casos el X forwarding viene configurado por defecto con el cliente ssh. Para deshabilitar esta propiedad usar:

Comandos 1,3:

```
$ ssh -x fulano@cluster.dominio
```

Este comando puede ser particularmente útil cuando la conexión al cluster se hace solamente para lanzar trabajos de computo.

Acceso desde Windows

Es también posible abrir una terminal remota usando un cliente ssh en Windows. Entre los clientes más populares, ligeros y versátiles para este sistema operativo se encuentran putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). En la figura 1 se muestra un pantallazo del programa. El uso de putty es bastante intuitivo.

Un tema de particular interés relacionado con la apertura de una terminal remota en Windows es el X forwarding, que como se explico antes permite a un usuario abrir desde la terminal remota una aplicación gráfica del frontend sobre la máquina en la que esta trabajando. 2 condiciones básicas deben satisfacerse para que se pueda hacer algo como esto: 1) la sesión de putty debe estar configurada para el X11 forwarding (ver figura 2), 2) la máquina windows en la que se esta trabajando debe estar corriendo un emulador del sistema de ventanas de Linux (X Windows).

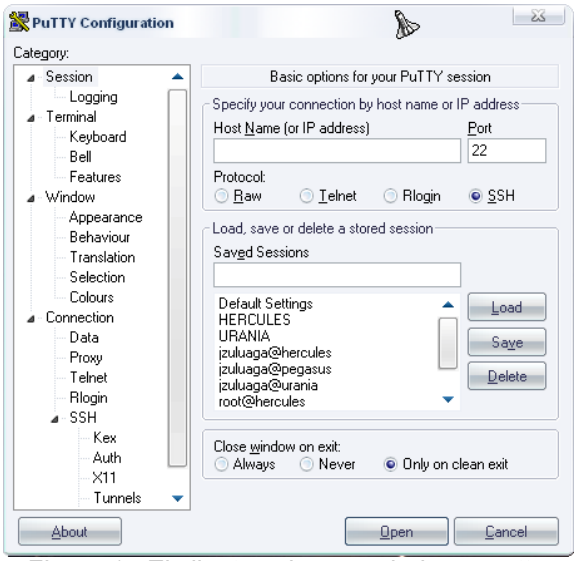


Figura 1. El cliente ssh para windows, putty.

La primera condición se cumple configurando la opción de X11 forwarding en el cliente putty y grabando esta elección en la sección que se esta configurando (ver figura 2).

La segunda condición requiere la instalación de un emulador de X Windows. Uno gratuito, muy ligero, sencillo de instalar y fácil de usar es el Xmins (http://www.iteisa.com/xmins).

Una vez configurado el cliente ssh y corriendo el emulador de X Windows podrán lanzarse programas gráficos del frontend para que corran sobre el modo gráfico de la terminal windows en la que se esta trabajando. Vale la pena aclarar que las aplicaciones que se lancen de este modo usaran siempre los recursos de computo del frontend.

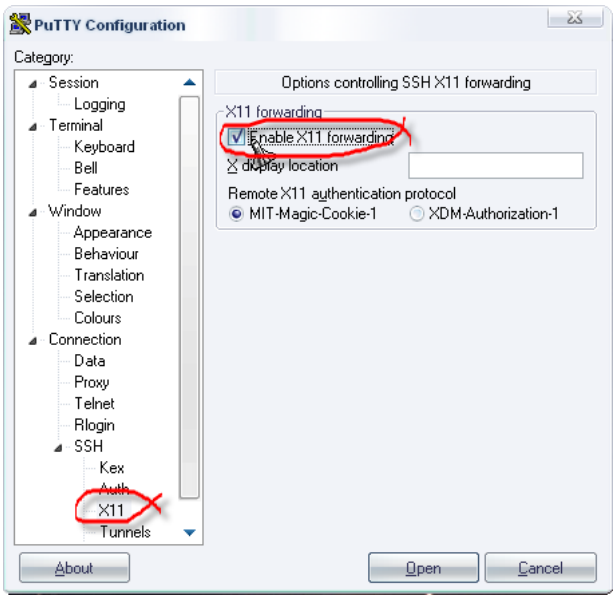


Figura 2. Configuración de putty para el X forwarding

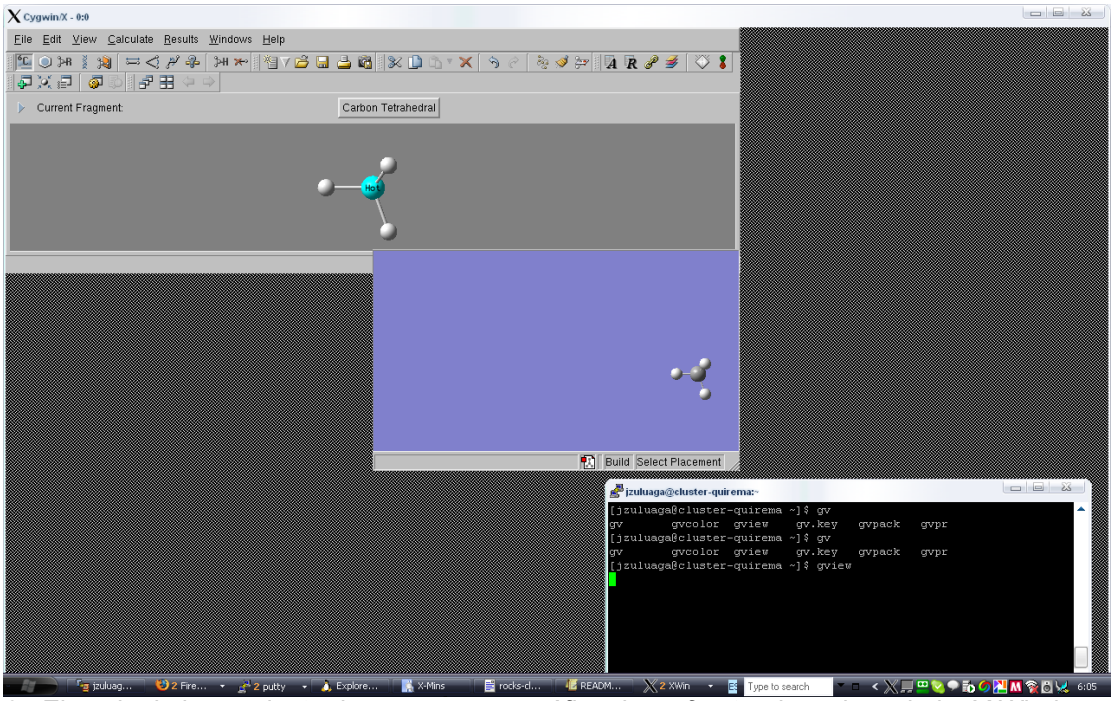


Figura 3. Ejemplo de lanzamiento de un programa gráfico de un frontend en el emulador X Windows de una terminal windows. La ventana abajo a la derecha es la terminal remota abierta con putty. El programa arriba a la izquierda es Gaussian View.

Tips útiles

Normalmente el administrador asignará a los usuarios una contraseña genérica para la

primera vez que se conecten con el frontend. Es muy importante sin embargo que una vez haya conseguido conectarse cambie inmediatamente su contraseña. Para cambiarla use el comando “passwd”:

Comandos 1,4:

$ passwd

```
Changing password for user course.
Changing password for course
(current) UNIX password:
New UNIX password:
BAD PASSWORD: is too similar to the old one
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Recuerde las recomendaciones generales para la creación de contraseñas validas y seguras en Linux: use más de 6 caracteres, mezcle letras, números y símbolos especiales, no use palabras del diccionario o una contraseña muy trivial, trate de asignar una contraseña que este utilizando en otros sistemas (correo electrónico, otras cuentas Linux, etc.)

Una vez en el frontend es posible acceder a cualquiera de los nodos de computo del cluster usando el mismo mecanismo de terminal remota ssh. Para hacerlo es necesario conocer el FQHN de los nodos del cluster o equivalentemente sus IPs.
El FQHN de los nodos del cluster es asignado automáticamente al momento de la instalación. Rocks utiliza la convención de llamar a los nodos usando el prefijo “compute-0” y el número de cada nodo. Los nodos están numerados comenzando en 0. Todos los nodos (y el frontend mismo) se encuentran inscritos en el dominio “.local”, solo reconocido en el cluster. Así finalmente los nombres de los nodos de computo del cluster serán compute-0-0.local, compute-0-1.local, compute-0-2.local, etc.

Tips útiles

Rocks define unos sobrenombres (aliases) para los nodos del cluster. Los aliases se construyen con el sufijo c0 y el número del cluster. Así los nodos c0-0, c0-1, c0-2 serán los mismos compute-0-0.local, compute-0-1.local, compute-0-2.local

La lista de todos los nodos del cluster con sus IPs, sus FQHN y sus aliases se encuentra en el archivo /etc/hosts:

/etc/hosts:

```
#
# Do NOT Edit (generated by dbreport)
#
127.0.0.1       localhost.localdomain   localhost       cluster
10.1.1.1        cluster.local   cluster #       originally      frontend-0-0
10.255.255.254  compute-0-0.local       compute-0-0     c0-0
10.255.255.253  compute-0-1.local       compute-0-1     c0-1
10.255.255.252  compute-0-2.local       compute-0-2     c0-2
10.255.255.251  compute-0-3.local       compute-0-3     c0-3
10.255.255.250  compute-0-4.local       compute-0-4     c0-4
10.255.255.249  compute-0-5.local       compute-0-5     c0-5
10.255.255.248  compute-0-6.local       compute-0-6     c0-6
10.255.255.247  compute-0-7.local       compute-0-7     c0-7
```

```
10.255.255.246  compute-0-8.local       compute-0-8     c0-8
192.168.3.2   cluster.dominio        cluster
```

Nótese que la IP del frontend es 10.1.1.1 y que las Ips de los nodos se asignan comenzando en 10.255.255.254 (c0-0) reduciendo por cada nodo el valor del último período de la IP en 1.

Para acceder a uno de los nodos del cluster se usa uno de los comandos:

Comandos 1,5:

$ ssh compute-0-4.local
$ ssh c0-4

```
Warning: Permanently added 'c0-4' (RSA) to the list of known hosts.
Rocks Compute Node
Rocks 4.2.1 (Cydonia)
Profile built 14:01 17-Oct-2006

Kickstarted 09:10 17-Oct-2006
[fulano@compute-0-4 ~]$
```

Tips útiles

Antes de intentar una conexión a uno de los nodos del cluster es interesante verificar que el nodo este arriba (encendido y recibiendo requerimientos de conexión). Esto se puede hacer usando el comando “ping”.

Comandos 1,6:

$ ping -c 2 compute-0-4.local

```
PING compute-0-6.local (10.255.255.248) 56(84) bytes of data.
64 bytes from compute-0-6.local (10.255.255.248): icmp_seq=0 ttl=64 time=0.143 ms
64 bytes from compute-0-6.local (10.255.255.248): icmp_seq=1 ttl=64 time=0.126 ms

--- compute-0-6.local ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.126/0.134/0.143/0.014 ms, pipe 2
```

El comando envia un paquete de Internet a la máquina remota y espera que el paquete sea devuelto. Además calcula el tiempo de ida y regreso del paquete (time). Si se obtiene este último valor la máquina esta arriba y recibiendo conexiones.

Es posible también hacer un ping a todos los nodos del cluster para verificar cuales están arriba y cuales abajo.

Comandos 1,7:

$ ping -b 10.0.0.0 -c 2 compute-0-4.local

```
WARNING: pinging broadcast address
PING 10.0.0.0 (10.0.0.0) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=0 ttl=64 time=0.064 ms
64 bytes from 10.255.255.249: icmp_seq=0 ttl=64 time=0.181 ms (DUP!)
64 bytes from 10.255.255.253: icmp_seq=0 ttl=64 time=0.186 ms (DUP!)
64 bytes from 10.255.255.254: icmp_seq=0 ttl=64 time=0.189 ms (DUP!)
64 bytes from 10.255.255.246: icmp_seq=0 ttl=64 time=0.199 ms (DUP!)
64 bytes from 10.255.255.248: icmp_seq=0 ttl=64 time=0.214 ms (DUP!)
```

```
64 bytes from 10.255.255.247: icmp_seq=0 ttl=64 time=0.225 ms (DUP!)
64 bytes from 10.255.255.251: icmp_seq=0 ttl=64 time=0.230 ms (DUP!)
64 bytes from 10.255.255.252: icmp_seq=0 ttl=64 time=0.239 ms (DUP!)
64 bytes from 10.255.255.250: icmp_seq=0 ttl=64 time=0.251 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.034 ms

--- 10.0.0.0 ping statistics ---
2 packets transmitted, 2 received, +9 duplicates, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.034/0.182/0.251/0.069 ms, pipe 2
```

Notese que el orden en el que responden no necesariamente coincide con el orden de las Ips. En la salida anterior puede verse también que el número de nodos de computo es 9 (véase "+9 duplicates").

1.2.Sistemas de archivos en el cluster

Los archivos que usa el usuario son almacenados automáticamente en su directorio casa(home directory). El directorio casa es normalmente /home/fulano (donde fulano es el login). En un sistema de computación en cluster es absolutamente necesario que esos mismos archivos puedan ser accedidos cuando se realiza una conexión remota a uno de los nodos de computo del cluster.

Rocks usa 2 mecanismos básicos para garantizar que el usuario tenga acceso directo a sus archivos aún si se encuentra conectado a uno de los nodos de computo. De un lado esta el NFS (Network filesystem) que "monta" el sistema de archivos de la cuenta del usuario a través de la red en el nodo al que se conecta; en el sistema NFS los cambios que se hacen sobre el sistema de archivos, sea en el frontend como en los nodos, se actualizan automáticamente en el disco duro del frontend donde residen realmente. El otro es el servicio autofs (Automounter) que garantiza que el "montado" de los sistemas de archivos sea automático y ocurra al momento de acceso del usuario; autofs también desmonta el sistema de archivos cuando el usuario deja de utilizarlo.

Para probar que el sistema funciona correctamente, mientras esta en el frontend cree un archivo vacío:

Comandos 1,8:

```
$ touch archivo-vacio-frontend
```

Ahora conéctelos a uno de los nodos y verifique que puede acceder al archivo que acaba de crear en el frontend desde ese nodo. De la misma manera, estando conectado con el nodo cree un archivo vacío allí.

Comandos 1,9:

```
$ ssh c0-3
$ touch archivo-vacio-nodo
```

Salga del nodo (comando exit) y verifique que puede ver el archivo que creo allí, en el directorio del usuario en el frontend.

Ejecución de comandos remotos en los nodos

Muchas veces es necesario ejecutar un comando simple en otro nodo sin necesidad de abrir una terminal remota en ese nodo. El comando ssh provee la herramienta básica para realizar este tipo de acciones remotas.

Por ejemplo para ejecutar el comando "w" en el nodo 4 (w muestra una lista de los usuarios conectados a una maquina linux) basta usar:

Comandos 1,10:

```
$ ssh c0-4 w
```

```
10:07:25 up  1:19,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
fulano   pts/0    192.168.3.2       09:20    0.00s  0.16s  0.06s ssh -x localhost
jzuluaga pts/1    localhost.locald  10:02    0.00s  0.03s  0.00s w
```

Debe tenerse presente que cuando se ejecutan comandos usando ssh, el directorio base del comando es el mismo directorio casa del usuario, no importa que el comando sea lanzado desde otro directorio.

Ejemplo:

Comandos 1,11:

```
$ mkdir directorio
$ cd directorio
$ ssh c0-0 pwd
```

El comando pwd informa el directorio base sobre el que se están ejecutando los comandos en linux.

Los comandos de la familia cluster

Uno de las más útiles herramientas proveídas por Rocks son los comandos de la familia cluster, cluster-fork, cluster-kill, cluster-ps y cluster-probe. Estos comandos permiten ejecutar "masivamente" comandos o acciones en todos los nodos del cluster. Esto es en lugar de usar ssh manualmente para pedir que un comando se ejecute en cada nodo del cluster se usa en su lugar cluster-fork por ejemplo. Así:

Comandos 1,12:

```
$ cluster-fork "w"
```

```
compute-0-0:
 10:12:18 up 11 days,  8:35,  0 users,  load average: 0.00, 0.01, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-1:
 10:12:19 up 16:25,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-2:
 10:12:23 up 8 days, 21:49,  0 users,  load average: 1.08, 1.03, 1.01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-3:
```

```
 10:12:27 up 7 days, 20:03,  0 users,  load average: 2.24, 2.05, 2.02
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-4:
 10:12:28 up 12 days, 16:20,  0 users,  load average: 1.00, 1.00, 1.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-5:
 10:12:29 up 12 days, 16:20,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-6:
 10:12:30 up 12 days, 16:20,  0 users,  load average: 1.08, 1.02, 1.01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-7:
 10:12:30 up 8 days, 22:03,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-8:
 10:12:31 up 12 days, 16:20,  0 users,  load average: 0.00, 0.01, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
```

Cluster fork utiliza internamente el comando ssh para la ejecución del comando en cada nodo del cluster. Se recomienda, como se muestra en el comando anterior, encerrar entre comillas el comando que se quiere ejecutar masivamente en el cluster.

Tips útiles

Es importante anotar que cuando la terminal sobre la que se esta trabajando tiene habilitado el X11 forwarding cada conexión ssh que se haga a un nodo usará también X11 forwarding. Esto hará mucho más lentas las conexiones y podría causar algunos dolores de cabeza cuando se ejecutan los comandos de la familiar cluster. Para evitar este comportamiento se recomienda “re-loggearse” en el frontend deshabilitando el X11 forwarding.

Comandos 1,13:

```
$ ssh -x localhost
```

Compare el tiempo de ejecución de un comando cluster-fork antes y después de ejecutar el anterior comando.

Cuando se ejecuta un comando de la familia cluster es posible “seleccionar” los nodos sobre los que deberá ejecutarse la acción o el comando seleccionado. Para hacerlo se debe usar:

Comandos 1,14:

```
$ cluster-fork -n “c0-2 c0-3 c0-5” w
```

Donde después de la opción -n y entre comillas se coloca la lista de los nodos separada por espacios en blanco.

Los comandos cluster-ps, cluster-probe y cluster-kill los conoceremos en su debido momento.

1.3.Monitoreo básico de los recursos del cluster

Antes de comenzar a calcular en el cluster se hace necesario conocer la disponibilidad y ocupación de los recursos de los que se dispone en la plataforma.

Existen en general 3 tipos de recursos que deben revisarse cuando se quiere usar el cluster para realizar trabajos de computo: 1) CPU: el uso y disponibilidad de tiempo CPU, 2) RAM: El uso de y disponibilidad de memoria RAM y memoria virtual, 3) HD: el uso y disponibilidad de disco duro.

El valor nominal e instantáneo de cada una de esas propiedades puede consultarse mediante distintos mecanismos disponibles en el sistema operativo Linux.

CPU

El uso de CPU es uno de los más importantes parámetros que deben monitorearse cuando se usa una maquina para realizar cálculos.

Estamos interesados en conocer 2 tipos de parámetros: parámetros estáticos (número de procesadores en la Mother Board, velocidad del reloj, tamaño de la memoria cache) y parámetros dinámicos (carga del procesador).

Parámetros estáticos

Para conocer los parámetros estáticos se debe consultar directamente el archivo /proc/cpuinfo:

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 3
model name      : Intel(R) Pentium(R) 4 CPU 3.20GHz
stepping        : 3
cpu MHz         : 3192.676
cache size      : 1024 KB
...

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 15
model           : 3
model name      : Intel(R) Pentium(R) 4 CPU 3.20GHz
stepping        : 3
cpu MHz         : 3192.676
cache size      : 1024 KB
...
```

En este archivo aparecerá una entrada por cada procesador “efectivo” que reconozca el sistema operativo. Debe anotarse que una máquina que use tecnología Hyperthreading parecerá tener 2 procesadores efectivos, aunque realmente solo tenga uno con capacidades para procesamiento paralelo.

Como puede verse en este archivo podemos conocer la velocidad del reloj de cada procesador (cpu Mhz) y el tamaño de la memoria cache (cache size).

Es posible consultar estos mismos parámetros estáticos de otros nodos del cluster usando el comando cluster-fork:

Comandos 1,15:

$ cluster-fork -n “c0-2 c0-3” “grep -A 8 processor /proc/cpuinfo”

```
c0-2:
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 3
model name      : Intel(R) Pentium(R) 4 CPU 3.20GHz
stepping        : 3
cpu MHz         : 3192.503
cache size      : 1024 KB
c0-3:
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      : Intel(R) Pentium(R) 4 CPU 3.60GHz
stepping        : 1
cpu MHz         : 3601.085
cache size      : 1024 KB
--
processor       : 1
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      : Intel(R) Pentium(R) 4 CPU 3.60GHz
stepping        : 1
cpu MHz         : 3601.085
cache size      : 1024 KB
```

De la salida del comando anterior vemos que en el nodo c0-2 de este cluster hay solamente un procesador mientras que en el nodo c0-3 se detectaron 2 procesadores idénticos (seguramente un procesador INTEL con HT).

Parámetros dinámicos

El parámetro dinámico más importante para conocer se conoce como la “carga promedio” del procesador (loadavg). La carga promedio del procesador se define de manera simple como el número efectivo de procesos que están (o estuvieron) en cola en un período de tiempo definido. Existen 3 loadavgs: la carga de los últimos 5, 10 y 15 minutos.

Existen distintas maneras para consultar en linux los loadavgs. El comando “uptime” ofrece una de las formas más expeditas:

Comandos 1,16:

$ uptime

```
 06:11:32 up 14 days, 21:27, 15 users,  load average: 0.25, 0.23, 0.22
```

El primer número indica la hora del reloj de la maquina, el segundo el tiempo que la maquina lleva encendida de forma continua, el tercer el número de usuarios que tienen una terminal abierta en este momento y los tres números finales indican la carga 5,10,15 del procesador (los parámetros que nos interesan).

Como una regla simple si una maquina tiene un número p de procesadores efectivos, el procesador se encontrará ocupado si el loadavg es aproximadamente p (para una máquina con un procesador si el loadavg es aproximadamente 1). Si el loadavg supera el número de procesadores efectivos entonces el procesador esta muy ocupado y no se recomienda cargarlo con más tareas.

Para consultar la carga promedio del procesador en otros nodos del cluster se puede usar:

Comandos 1,17:

$ cluster-fork “uptime”

```
compute-0-0:
 06:15:58 up 31 days, 19:12,  0 users,  load average: 1.33, 1.48, 1.46
compute-0-1:
 06:15:58 up 12 days, 14:25,  0 users,  load average: 0.13, 0.05, 0.02
compute-0-2:
 06:15:58 up 31 days, 19:12,  0 users,  load average: 0.09, 0.03, 0.00
compute-0-3:
 06:15:59 up 31 days, 19:12,  0 users,  load average: 0.04, 0.03, 0.00
compute-0-4:
 06:15:59 up 12 days, 14:23,  0 users,  load average: 0.15, 0.07, 0.01
compute-0-5:
 06:15:59 up 31 days, 19:12,  0 users,  load average: 0.07, 0.03, 0.00
compute-0-6:
 06:16:00 up 31 days, 19:12,  0 users,  load average: 0.01, 0.00, 0.00
compute-0-7:
 06:16:00 up 12 days, 14:23,  0 users,  load average: 1.02, 1.02, 1.00
compute-0-8:
 06:16:00 up 31 days, 19:12,  0 users,  load average: 0.02, 0.04, 0.00
```

RAM

De nuevo en este caso podemos determinar parámetros estáticos (tamaño total de memoria RAM, tamaño total de memoria swap) y parámetros dinámicos (% de memoria RAM utilizado, % de memoria swap utilizada).

Parámetros estáticos

Para conocer los parámetros estáticos asociados con la memoria RAM se debe consultar directamente el archivo /proc/meminfo:

```
MemTotal:      1017584 kB
MemFree:        684860 kB
...
SwapTotal:     1020116 kB
SwapFree:      1020116 kB
...
```

De resaltar entre la enorme cantidad de parámetros reportada en este archivo esta la cantidad total de memoria RAM con la que cuenta la maquina (MemTotal) y la cantidad de memoria swap total disponible (SwapTotal).

Es posible consultar estos mismos parámetros estáticos de otros nodos del cluster usando el

comando cluster-fork:

Comandos 1,18:

$ cluster-fork -n “c0-0 c0-1” “grep -A 1 MemTotal /proc/cpuinfo”

```
c0-0:
MemTotal:      1017584 kB
MemFree:        855796 kB
c0-1:
MemTotal:      1017584 kB
MemFree:        382308 kB
```

Comandos 1,19:

$ cluster-fork -n “c0-0 c0-1” “grep -A 1 SwapTotal /proc/cpuinfo”

```
c0-0:
SwapTotal:     1020116 kB
SwapFree:       993136 kB
c0-1:
SwapTotal:     1020116 kB
SwapFree:      1020116 kB
```

Parámetros dinámicos

El único parámetro dinámico de interés relacionado con el uso de la memoria RAM es la cantidad de RAM o swap disponible en la maquina en un determinado momento. Se puede consultar este parámetro de la misma manera como se hizo anteriormente.

HD

Los parámetros estáticos del disco incluyen el número de particiones de disco, el directorio de montaje del sistema de archivos de cada partición y el tamaño de las particiones. Los parámetros dinámicos incluyen la partición en el que se escriben los datos de un determinado programa y el espacio disponible en esa partición.

Parámetros estáticos

Para consultar el número de particiones, su punto de montaje y el tamaño total de esas particiones es posible usar el comando “df”:

Comandos 1,20:

$ df -h

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             7.7G  3.7G  3.7G  51% /
none                  497M     0  497M   0% /dev/shm
/dev/hda5              63G   12G   49G  19% /state/partition1
/dev/hda2             3.9G  301M  3.4G   9% /var
tmpfs                 243M  4.5M  239M   2% /var/lib/ganglia/rrds
```

Notese que de las particiones que presenta al comando df incluyen sistemas de archivos que no son de interés directo para el usuario (/dev/shm, /var/lib/ganglia/rds, etc.)
Solo las particiones del disco /dev/hda* son de interés. De ellas vale la pena resaltar la

partición /state/partition1 del frontend que contiene los home directory de los usuarios. El tamaño de esa partición define la capacidad total para almacenamiento de información de los usuarios. Además en los nodos esa misma partición contiene a veces el espacio de almacenamiento temporal de algunas aplicaciones (por ejemplo el scratch del programa Gaussian) y por lo tanto su capacidad definirá también la disposición que esas aplicaciones tienen de espacio.

De nuevo es posible consultar el tamaño total de las particiones de los nodos de computo del cluster usando cluster-fork.

Comandos 1,21:

$ cluster-fork -n “c0-0 c0-1” “df -h”

```
compute-0-0:
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             7.7G  2.9G  4.5G  39% /
none                  497M     0  497M   0% /dev/shm
/dev/hda5              61G  7.8G   51G  14% /state/partition1
/dev/hda2             3.9G   75M  3.6G   2% /var
cluster-quirema.local:/export/home/jforrego
                       63G   12G   49G  19% /home/jforrego
cluster-quirema.local:/export/home/.gaussian03
                       63G   12G   49G  19% /home/.gaussian03
compute-0-1:
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             7.7G  2.9G  4.5G  39% /
none                  497M     0  497M   0% /dev/shm
/dev/hda5              63G  173M   60G   1% /state/partition1
/dev/hda2             3.9G   74M  3.6G   2% /var
cluster-quirema.local:/export/home/solmi
                       63G   12G   49G  19% /home/solmi
cluster-quirema.local:/export/home/.gaussian03
                       63G   12G   49G  19% /home/.gaussian03
```

Notese en la salida anterior como la partición del usuario fulano y del programa Gaussian aparecen montadas en la máquina c0-1. Este montaje lo hacen los servicios NFS y autofs mencionados más arriba. Es de aclarar que el tamaño de la partición reportado en este último caso hace referencia a una partición del frontend directamente y no del nodo de computo. Las particiones del nodo son las que aparecen referidas directamente a su disco duro, /dev/hda1, /dev/hda5, etc.

Parámetros dinámicos

En este caso el parámetro dinámico sensible de requerirse es la cantidad de espacio disponible en las particiones críticas (partición de calculo) etc. en los nodos de computo del cluster. Vale la pena aclarar a este punto que si los programas que se corren solo utilizan como espacio de almacenamiento el home directory entonces solamente tendrá interés conocer el espacio disponible en la partición /state/partition1 del frontend.

Monitoreo de Procesos con ps

Otra herramienta de gran valor en el monitoreo del cluster la constituye el comando “ps” para monitoreo de procesos en Linux.

ps permite determinar los procesos en ejecución en una máquina linux y las propiedades específicas de esos procesos. El uso detallado de ps es un tema complejo y que va más allá de esta guía práctica. A continuación se ofrece una guía básica para el monitoreo de procesos usando opciones básicas de este comando.

Comandos 1,22:

$ ps caux

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0   2868   552 ?        S    08:48   0:00 init
root          2  0.0  0.0      0     0 ?        S    08:48   0:00 migration/0
root          3  0.0  0.0      0     0 ?        SN   08:48   0:00 ksoftirqd/0
root          4  0.0  0.0      0     0 ?        S    08:48   0:00 migration/1
root          5  0.0  0.0      0     0 ?        SN   08:48   0:00 ksoftirqd/1
root          6  0.0  0.0      0     0 ?        S<   08:48   0:00 events/0
root          7  0.0  0.0      0     0 ?        S<   08:48   0:00 events/1
root          8  0.0  0.0      0     0 ?        S<   08:48   0:00 khelper
root          9  0.0  0.0      0     0 ?        S<   08:48   0:00 kacpid
...
```

Muestra todos los procesos en ejecución en una máquina. La lista de procesos que produce este comando puede llegar a ser muy grande y se hace necesaria filtrarla usando determinados criterios. Así por ejemplo para conocer los procesos que esta ejecutando el usuario jzuluaga puede usarse:

Comandos 1,24:

$ ps caux | grep jzuluaga

```
jzuluaga  4408  0.0  0.2  9732 4628 ?        S    Oct17   0:02 gconfd-2
jzuluaga 16781  0.0  0.0 36616  832 ?        S    Oct26   0:00 basicroutines.o
jzuluaga 16782  0.0  0.0  4412 1836 ?        S    Oct26   0:00 ssh
jzuluaga 22095  0.0  0.1  8612 2652 ?        S    Oct31   0:07 sshd
jzuluaga 22096  0.0  0.0  6380 1476 pts/0    Ss+  Oct31   0:00 bash
jzuluaga 22147  0.0  0.5 35400 11908 pts/0   Sl   Oct31   0:03 gnome-terminal
jzuluaga 22149  0.0  0.1  7484 2528 ?        Ss   Oct31   0:00 bonobo-activati
jzuluaga 22150  0.0  0.0  2840  616 pts/0    S    Oct31   0:00 gnome-pty-helpe
jzuluaga 22151  0.0  0.0  6032 1492 pts/1    Ss+  Oct31   0:00 bash
jzuluaga 22396  0.0  0.3 14000 7484 pts/1    S    Oct31   0:01 emacs
jzuluaga 22412  0.0  0.4 13772 8444 pts/1    S    Oct31   0:02 emacs
jzuluaga 22580  0.0  0.0  4668 1448 pts/3    Ss   Oct31   0:00 bash
jzuluaga 22646  0.0  0.0  4356 1852 pts/3    S+   Oct31   0:00 ssh
jzuluaga 22649  0.0  0.1  7632 2196 ?        S    Oct31   0:00 sshd
jzuluaga 22650  0.0  0.0  6260 1484 pts/5    Ss+  Oct31   0:00 bash
jzuluaga 23776  0.0  0.4 15568 8852 pts/1    S    Oct31   0:04 emacs
```

La información proveída por el comando ps con la opción “u” es: el usuario que corre el proceso, el número del proceso (PID), el % de CPU utilizada, el % de la memoria utilizada, la cantidad de memoria virtual (en kbytes) usada (VSZ), la cantidad de memoria RAM utilizada (en kbytes) (RSS), el estado del proceso, la fecha de inicio, la cantidad de tiempo que ha estado en ejecución y el comando abreviado asociado al proceso.

Es posible cambiar la información desplegada por ps usando la opción “-o” seguida de los campos que se quieren mostrar:

Comandos 1,25:

$ ps a -o pid,user,cmd | grep jzuluaga

```
jzuluaga 00:00:00 -bash
jzuluaga 00:00:03 gnome-terminal
jzuluaga 00:00:00 gnome-pty-helper
jzuluaga 00:00:00 bash
jzuluaga 00:00:01 emacs Makefile
jzuluaga 00:00:02 emacs data-preparation-pack.c data-preparation-struct.c
jzuluaga 00:00:00 bash
jzuluaga 00:00:00 ssh -x localhost
jzuluaga 00:00:00 -bash
jzuluaga 00:00:04 emacs data-preparation-pack.c
root     00:00:00 grep jzuluaga
```

En este caso se ha pedido mostrar todos los procesos que corre jzuluaga pero indicando solamente el PID, el nombre del usuario y el nombre completo del comando (incluyendo las opciones.

Las cadenas validas que se pueden usar con la opción -o incluyen: pid, user, cmd, %cpu, %mem, etime (tiempo que lleva ejecutándose el proceso), comm (nombre abreviado del comando), lstart (comienzo del proceso), rss (cantidad de memoria RAM en uso).

Particularmente interesante en el cluster es el comando:

Comandos 1,26:

$ ps r -A

Que muestra los procesos en ejecución en una máquina. Para mostrar todos los procesos en ejecución en el cluster se usaría un comando como el siguiente:

Comandos 1,27:

$ cluster-fork -n “c0-0 c0-1” “ps r -A -o user,%cpu,%mem,etime,comm”

```
c0-0:
USER     %CPU %MEM     ELAPSED COMMAND
root      0.0  0.0       00:00 ps
c0-1:
USER     %CPU %MEM     ELAPSED COMMAND
eflorez  99.9 17.9    01:49:42 l502.exe
root      3.0  0.0       00:01 ps
```

1.4.Síntesis de comandos

Comando	Explicación
$ ssh fulano@cluster.dominio	Conexión al cluster con el cliente ssh de Linux
$ ssh fulano@192.168.3.2	
$ ssh -XY fulano@cluster.dominio	Conexión con el cliente ssh de Linux usando X11 forwarding
$ ssh -x fulano@cluster.dominio	Conexión con el cliente ssh de Linux sin usar X11 forwarding
$ passwd	Comando para cambio de contraseña en Linux
$ ssh compute-0-4.local	Comando de conexión por ssh a uno de los nodos del cluster
$ ssh c0-4	
$ ping -c 2 compute-0-4.local	Verificación de la conectividad con uno de los nodos del cluster
$ ping -b 10.0.0.0 -c 2 compute-0-4.local	Verificación de la conectividad con todos los nodos del cluster
$ touch archivo-vacio-frontend	Creación de un archivo en blanco
$ ssh c0-4 w	Ejecución remota en uno de los nodos del comando w para ver los usuarios loggeados en el nodo
$ ssh c0-0 pwd	Ejecución remota del comando pwd para saber el directorio de trabajo
$ cluster-fork “w’’	Es recomendable acostumbrarse a escribir el comando de cluster-fork entre comillas
$ cluster-fork -n “c0-2 c0-3 c0-5” w	Ejecución de un comando en varios nodos del cluster, escogiendo los nodos
$ cluster-fork -n “c0-2 c0-3” “grep -A 8 processor /proc/cpuinfo”	Comando para la determinación de las propiedades estáticas de la CPU de algunos nodos
$ uptime	Determinación de las propiedades dinámicas de la CPU en Linux
$ cluster-fork “uptime”	Determinación de las propiedades dinámicas de la CPU en todos los nodos del cluster
$ cluster-fork -n “c0-0 c0-1” “grep -A 1 MemTotal /proc/cpuinfo”	Consulta de las propiedades estáticas de la memoria RAM en el cluster
$ cluster-fork -n “c0-0 c0-1” “grep -A 1 SwapTotal /proc/cpuinfo”	Consulta de las propiedades estáticas de la memoria swap en el cluster
$ df -h	Determinación de las propiedades de los sistemas de archivos montados en Linux
$ cluster-fork -n “c0-0 c0-1” “df -h”	Determinación de las propiedades de los sistemas de archivos montados en el cluster
$ ps caux	Muestra todos los procesos en la máquina
$ ps caux | grep jzuluaga	Muestra todos los procesos en la máquina pertenecientes al usuario jzuluaga

Comando	Explicación
$ ps a -o pid,user,cmd | grep jzuluaga	Muestra todos los procesos de jzuluaga indicando en la salida solamente el identificador del proceso, el nombre del usuario y el nombre completo del comando
$ ps r -A	Muestra todos los procesos en ejecución
$ cluster-fork -n “c0-0 c0-1” “ps r -A -o user,%cpu,%mem,etime,comm”	Muestra todos los procesos en ejecución en algunos nodos del cluster mostrando el usuario, el % de cpu utilizado, el % de ram, el tiempo que lleva en ejecución y el nombre sintético del comando.

Primera Parte
Guía Práctica del Usuario

Guía 2
Calculando con el Cluster

Contenido sintético

2.Calculando con en el cluster

2.1.Lanzamiento y monitoreo de un trabajo en el cluster

El lanzamiento de trabajos de computo en el cluster podría hacerse de manera tan sencilla como identificar mediante los procedimientos explicados en el documento 1 un nodo o varios con la suficiente disponibilidad de carga, conectarse vía ssh a esos nodos y ejecutar en ellos el programa o programas que se quieren ejecutar.

Este tipo de procedimiento sin embargo inconvenientes y va en contravía del uso apropiado de la plataforma. Entre otras, se exponen aquí algunas razones por las que deben evitarse este tipo de prácticas en el cluster: 1) el uso del cluster podría no ser balanceado en tanto los usuarios determinan con sus propios criterios cuál o cuáles son los nodos disponibles y escogen generalmente los primeros nodos o los más poderosos para ejecutar sus trabajos. De ese modo gran parte del recurso podría verse sutilizada o inutilizada del todo. 2) No se respeta un orden de ejecución ni unas prioridades para los usuarios en tanto el uso los recursos se asignarían al primero que los solicitará o usará en lugar de asignarlos de acuerdo a un tiempo de espera por esos recursos. 3) Se consume normalmente un tiempo innecesario en la búsqueda de recursos disponibles para la ejecución de un trabajo. Esta es una tarea que no debería estar a cargo del usuario para el que la plataforma debería funcionar como un computador único al que entrega uno o muchos trabajos y el devuelve unos resultados. 4) Se puede desaprovechar las oportunidades de concurrencia en la ejecución de muchos procesos. 5) La ejecución de un programa desde la consola o incluso si el programa corre en background presenta inconvenientes. La interrupción accidental de la sesión del shell o del proceso en background podría arruinar el trabajo de calculo de horas.

Con todas estos inconvenientes la recomendación siempre es dejar la ejecución de los procesos a sistemas de programación de procesos (Job Schedulers) u otros sistemas de colas disponibles en el sistema operativo (queue systems).

En Rocks existen varios sistemas de programación de procesos que pueden usarse para

lanzar procesos en los nodos de forma controlada. Entre los más importantes identificamos el sistema de colas at o batch (incorporado directamente en el sistema operativo Linux), SGE (Sun Grid Engine) un sencillo pero muy completo programador de trabajos en el cluster, Condor, probablemente el más completo y robusto programador de trabajos open source y PBS, otro programador de trabajos muy popular entre los usuarios de plataformas distribuidas. En esta guía práctica nos concentraremos en el uso de SGE, por su relativa simplicidad en el uso y al mismo tiempo por proveernos de herramientas poderosas para la administración y monitoreo de los procesos.

En lo sucesivo asumiremos que el usuario necesita ejecutar el programa **program.out**, usando como entradas el archivo **input.dat** que debe pasarse como un parámetro por línea de comandos al programa. Además el programa utiliza el archivo **parameters.param** que debe leer al momento de la ejecución. La salida del programa va dirigida una parte a la pantalla (standard output) y otra a la sálida de error (standard error). Además el programa produce un archivo de salida **output.dat**. Como una condición adicional asumiremos que para ejecutarse el programa necesita que se fije el valor de la variable de ambiente del sistema **PROGRAMTMP** donde se indica el directorio que usa el programa para almacenar información temporal.

El directorio donde residen el binario y los archivos relacionados con el programa tiene el siguiente contenido:

```
total 16
-rw-rw-r--  1 jzuluaga jzuluaga     0 Nov  3 07:27 input.dat
-rw-rw-r--  1 jzuluaga jzuluaga    14 Nov  3 07:22 parameters.param
-rwxrwxr-x  1 jzuluaga jzuluaga 12283 Nov  3 07:27 program.out
```

Para ejecutar de manera normal el programa se usan los comandos:

Comandos 2,1:

```
$ export PROGRAMTMP=/tmp
$ ./program.out input.dat
```

```
Particle 1, time 1.000000...
Time elapsed 1.000000
Particle 2, time 1.000000...
Time elapsed 1.000000
Particle 3, time 6.000000...
Time elapsed 6.000000
Particle 4, time 8.000000...
Time elapsed 8.000000
Particle 5, time 12.000000...
Time elapsed 12.000000
...
```

Al concluir el programa el directorio de ejecución debe contener un archivo nuevo de nombre output.dat.

La ejecución de un programa usando SGE se realiza en 3 etapas: 1) preparación de un archivo de lanzamiento (launching script), 2) lanzamiento del proceso (submission), 3) monitoreo del proceso, 4) finalización, evaluación de la ejecución y recolección de los resultados. A continuación explicamos de forma práctica y sintética cada una de estas etapas.

Preparación del archivo de lanzamiento

El lanzamiento de cualquier proceso en SGE se realiza desde un programa del shell (shell script). A continuación se presenta el script de lanzamiento para el programa de ejemplo (se usan aquí las opciones más sencillas para el lanzamiento del proceso):

launch-program.sh:

```
#!/bin/bash
#Launching script for program.out
#Submission options:
#$-S /bin/bash
#$-N program
#$-cwd

export PROGRAMTMP=/tmp
./program.out input.dat
```

Todas las líneas en el script que comienzan con el símbolo "#" se tratan como comentarios del script, excepto aquellas que comienzan con "#$" que corresponden a las opciones de lanzamiento de SGE. Entre las opciones utilizadas en este sencillo script identificamos:

-S: Con esta opción se indica el interprete que debe usarse para ejecutar el script de lanzamiento. En este caso el script tiene la sintaxis de bash y por lo tanto el interprete de este lenguaje será el que utilicemos.

-N: Esta opción permite indicar el nombre del proceso. El nombre es arbitrario y asignado por el usuario. Debe evitarse el uso de espacios en blanco, caracteres especiales o protegidos (,;:#, etc.)

-cwd: esta es una importante opción que hace que el proceso que se esta lanzando utilice como directorio base el directorio desde el que se lanza y no el directorio casa que es el directorio base por defecto. Si no se usa la opción -cwd en el script se deberán realizar todas las acciones necesarias para ubicarse sobre el directorio de ejecución de modo que los archivos de entrada puedan encontrarse.

Lanzamiento del proceso

El lanzamiento del proceso se realiza mediante el comando:

Comandos 2.2:

```
$ qsub launch-program.sh
```

```
Your job 17 ("program") has been submitted
```

Notese que el programador de trabajos asigna al proceso un número secuencial (17) que deberá conservarse para solicitar más adelante información sobre ese mismo proceso.

Monitoreo del proceso

Para verificar que el proceso esta efectivamente en cola de ejecución se usa el comando:

Comandos 2,3:

$ qstat

```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
     17 0.55500 program    jzuluaga     r     11/02/2006 07:22:52 all.q@compute-0-4.local           1
```

La información desplegada por qstat incluye: job-ID, identificador asignado al proceso por el programador; prior, un número que indica la prioridad del proceso; el nombre asignado al proceso por el programador o por el usuario a través de la opción -N;user;state, el estado del proceso (r: run, q: en cola, w: waiting, E: con errores); submit/start at, el tiempo de lanzamiento y el inicio de la ejecución;queue, la cola en la que se esta ejecutando el proceso; slots, el número del procesador del nodo en el que se esta ejecutando el proceso; ja-task-ID, la posición del proceso en un arreglo de procesos (en este caso solo se esta lanzando un proceso y esta variable no esta definida.

Para verificar su estado de ejecución se puede usar también el comando qstat o enriquecerlo con la opción -f que presenta información detallada sobre todas las colas de trabajos presentes en el cluster:

Comandos 2,4:

$ qstat -f

```
queuename                      qtype used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   0/1       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   1/2       0.00     lx26-x86
     17 0.55500 program    jzuluaga     r     11/02/2006 07:22:52     1
---------------------------------------------------------------------------------
all.q@compute-0-5.local        BIP   0/2       0.01     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-6.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-7.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-8.local        BIP   0/2       0.01     lx26-x86
```

Notese que por cada nodo aparece información útil sobre sus propiedades y el estado de los procesos que se ejecutan en él. La información desplegada por este comando incluye: queuename, el nombre de la cola, que por defecto tiene la forma all.q@<nodo>;qtype, el tipo de cola (B:Batch, I:interactive, P:parallel); used/tot, el número de procesadores efectivos usados y disponibles en el nodo; load_avg, la carga del procesador en los últimos 5 minutos; arch, la arquitectura del nodo; states, el estado del nodo (en caso de existir un estado anormal).

Finalización, evaluación y recolección de resultados

Una vez finalizada la ejecución del proceso, es posible evaluar los resultados de la ejecución solicitando información detallada sobre la tarea al mismo programador. En SGE esto se realiza usando el comando qacct:

Comandos 2,5:

$ qacct -j 17

```
==============================================================
qname        all.q
hostname     compute-0-4.local
group        users
owner        jzuluaga
project      NONE
department   defaultdepartment
jobname      program
jobnumber    17
taskid       undefined
account      sge
priority     0
qsub_time    Thu Nov  2 07:22:41 2006
start_time   Thu Nov  2 07:22:52 2006
end_time     Thu Nov  2 07:24:58 2006
granted_pe   NONE
slots        1
failed       0
exit_status  0
ru_wallclock 126
ru_utime     121
...
cpu          126
mem          93.296
io           0.000
iow          0.000
maxvmem      768.391M
```

De la información presentada arriba pueden extraerse datos de gran interés sobre la ejecución del programa: en que nodo fue ejecutado el proceso, en que procesador de ese nodo fue ejecutado (slots), cuando comenzó y cuando término, cuanto tiempo tomo la ejecución en segundos (cpu) y cuánta memoria ram requirió (mem).

Una vez concluido el proceso el directorio de ejecución contiene los siguientes archivos:

Comandos 2,6:

$ ls -l

```
total 28
-rw-r--r--  1 jzuluaga users    0 Nov  2 07:17 input.dat
-rw-r--r--  1 jzuluaga users  150 Nov  2 07:17 launch-program.sh
-rw-r--r--  1 jzuluaga users 2505 Nov  2 07:24 output.dat
-rw-r--r--  1 jzuluaga users   14 Nov  2 07:17 parameters.param
-rw-r--r--  1 jzuluaga users 1158 Nov  2 07:24 program.e17
-rw-r--r--  1 jzuluaga users 1549 Nov  2 07:24 program.o17
-rwxr-xr-x  1 jzuluaga users 7737 Nov  2 07:22 program.out
```

Allí efectivamente comprobamos la aparición del archivo de salida output.dat que garantiza que el programa fue ejecutado efectivamente. Adicionalmente encontramos dos archivos nuevos program.o17 y program.e17. El contenido de estos archivos se presenta a continuación:

program.o17:

```
Particle 1, time 7.000000...
```

```
Particle 2, time 9.000000...
Particle 3, time 9.000000...
Particle 4, time 10.000000...
Particle 5, time 11.000000...
Particle 6, time 18.000000...
Particle 7, time 20.000000...
...
```

program.e17:

```
Time elapsed 7.000000
Time elapsed 9.000000
Time elapsed 9.000000
Time elapsed 10.000000
Time elapsed 11.000000
Time elapsed 18.000000
Time elapsed 20.000000
...
```

Allí comprobamos que el archivo ".o" contiene la sálida en pantalla (standard output) mientras que el archivo ".e" contiene la salida de error (standard error). Estos archivos se diferencian de otros archivos de salida y de error con el número de proceso que asigno SGE a la tarea.

Es posible enriquecer aún mas cada una de las etapas descritas anteriormente. A continuación se ofrecen algunos tips de interés para personalizar aún mas este proceso.

Tips útiles

Preparación del archivo de lanzamiento:

Otras opciones de interés que pueden incluirse en el archivo de lanzamiento incluyen:

-o: Permite escoger arbitrariamente el nombre del archivo para el standard output.
-e: Permite escoger arbitrariamente el nombre del archivo para el standard error.
-j y: Hace que el standard output y el standard error se pongan en el mismo archivo.
-a CCYYMMDDhhmm.ss: Permite escoger el tiempo en el que el proceso efectivamente se pondrá en cola de ejecución. Esta opción es útil por ejemplo cuando se desea programar la ejecución de procesos en horarios descongestionados.
-m [b|e]: Envía un correo electrónico al usuario al principio (b) o al final de la ejecución del proceso (e).
-now y: Se utiliza cuando se desea que el programador intente enviar inmediatamente el proceso a ejecución y no que lo coloque en cola. En caso de usar esta opción cuando se lance el proceso con qsub la ejecución del comando solo finalizará cuando el proceso se haya asignado a un procesador.
-q <cola1>,<cola2>,...: Permite escoger una lista de colas en las que el proceso debe ser ejecutado. Esta opción es particularmente útil si se quiere ejecutar uno o varios procesos en un nodo seleccionado. El nombre de las colas debe describirse usando la regla all.q@<nodo>.

Otras opciones pueden encontrarse consultando la página de manual del comando qsub:

Comandos 2.7:

```
$ man qsub
```

A continuación se muestra un archivo de lanzamiento que usa un conjunto más rico de opciones:

launch-program-richer.sh:

```
#!/bin/bash
#Launching script for program.out
#Submission options:
#$-S /bin/bash
#$-N program
#$-o job_output.$JOB_ID
#$-j y
#$-q all.q@compute-0-0.local
#$-a 200611032300
#$-m b
#$-cwd

export PROGRAMTMP=/tmp
./program.out input.dat
```

Nótese en este caso el uso de la variable de ambiente especial JOB_ID que almacena el número del proceso asignado por el programador y que permite nombrar de manera diferenciada el archivo de salida que además por la opción "-j y" contendrá tanto la standard output y la standard error.

Lanzamiento del proceso:

Todas las opciones que se indicaron en el parágrafo anterior pueden pasarse al comando qsub cuando se hace el lanzamiento del programa. Así por ejemplo el comando:

Comandos 2,8:

$ qsub -now y launch-program.sh

```
Your job 20 ("program") has been submitted
Waiting for immediate job to be scheduled.

Your immediate job 20 has been successfully scheduled.
```

Obligará al programador de procesos a enviar inmediatamente el programa a ejecución.

Monitoreo del proceso:

Durante la ejecución del proceso es posible interrumpir o detener completamente el proceso. La interrupción completa de un proceso se realiza usando el comando qdel:

Comandos 2,9:

$ qdel 21

```
jzuluaga has registered the job 21 for deletion
```

Una vez detenido el proceso no podrá continuar más.

2.2.Lanzamiento de múltiples trabajos en el cluster

Hasta este punto la ventaja que ofrece el uso de programadores de trabajo en el cluster es la de dejar a un sistema automático la búsqueda y elección de un nodo para la ejecución de un trabajo, poniendo el proceso en una cola en donde se respeta un orden y una prioridad de ejecución, una condición importante para el trabajo en una comunidad de usuarios.

Sin embargo el potencial más importante de una plataforma como el cluster esta en la posibilidad de ejecutar en forma concurrente un conjunto de procesos de modo que el resultado de por ejemplo un estudio paramétrico (ejecución del mismo programa con distintos parámetros de entrada) muy grande puede realizarse en muy poco tiempo.

Existen 2 formas básicas en las que pueden lanzarse un conjunto de procesos concurrentes (más no relacionados en tiempo de ejecución) en el cluster usando SGE.

Lanzamiento independiente de múltiples procesos

La forma más obvia para lanzar múltiples procesos concurrentes en el cluster consiste en crear un directorio por cada uno de los procesos que se desea lanzar y copiar en esos directorios los archivos necesarios para la ejecución.

Así por ejemplo:

Comandos 2,10:

```
$ mkdir run{1,2,3}
$ cp program.out input.dat parameters.param launch-program.sh run1
$ cp program.out input.dat parameters.param launch-program.sh run2
$ cp program.out input.dat parameters.param launch-program.sh run3
```

Creará 3 directorios para ejecuciones independientes del programa y copiará en ellos los archivos necesarios para esa ejecución. A continuación para diferenciar las 3 ejecuciones distintas pueden cambiarse los valores del archivo de parámetros (parameters.param) y del archivo de entrada (input.dat).

Una vez los archivos en cada directorio se diferencian se puede lanzar los procesos usando qsub:

Comandos 2,11:

```
$ cd run1;qsub launch-program.sh;cd ..
$ cd run2;qsub launch-program.sh;cd ..
$ cd run3;qsub launch-program.sh;cd ..
$ qstat -f
```

```
queuename                      qtype used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   1/2       0.00     lx26-x86
     32 0.55500 program    jzuluaga     r     11/02/2006 08:49:20     1
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   0/1       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   1/2       0.00     lx26-x86
```

```
     33 0.55500 program    jzuluaga     r     11/02/2006 08:49:20     1
----------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   0/2       0.91     lx26-x86
----------------------------------------------------------------------------
all.q@compute-0-5.local        BIP   0/2       0.01     lx26-x86
----------------------------------------------------------------------------
all.q@compute-0-6.local        BIP   0/2       0.03     lx26-x86
----------------------------------------------------------------------------
all.q@compute-0-7.local        BIP   0/2       0.01     lx26-x86
----------------------------------------------------------------------------
all.q@compute-0-8.local        BIP   1/2       0.06     lx26-x86
     31 0.55500 program    jzuluaga     r     11/02/2006 08:49:05     1
```

Lanzamiento de un arreglo de procesos

Es posible también lanzar en un solo comando múltiples procesos en el cluster. Se habla en este caso del lanzamiento de un arreglo de procesos. Debe tenerse en cuenta en este caso sin embargo que los archivos de entrada, parámetros y sálida deberán diferenciarse usando mecanismos propios del programa que se va a ejecutar.

Así por ejemplo si se quiere ejecutar un arreglo de 3 procesos con archivos de entrada input_a.dat, input_b.dat e input_c.dat, puede prepararse un archivo de lanzamiento como el que se muestra a continuación:

launch-program-multiple.sh:

```
#!/bin/bash
#Launching script for program.out
#Submission options:
#$-S /bin/bash
#$-N program
#$-cwd
#$-t 1-3:1

export PROGRAMTMP=/tmp
inputs=([1]=input_a.dat [2]=input_b.dat [3]=input_c.dat)

./program.out ${inputs[$SGE_TASK_ID]}
```

Notese el uso de la variable ambiente especial SGE_TASK_ID que distingue unos procesos de otros en el arreglo de procesos que se esta lanzando.

El lanzamiento de un arreglo de procesos se hace de forma convencional:

Comandos 2,12:

$ qsub launch-program-multiple.sh

```
Your job-array 35.1-3:1 ("program") has been submitted
```

El monitoreo del proceso con el comando qstat muestra información adicional respecto al caso del lanzamiento de un único proceso:

Comandos 2,13:

$ qstat

```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50 all.q@compute-0-1.local            1 3
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50 all.q@compute-0-2.local            1 2
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50 all.q@compute-0-6.local            1 1
```

Notese que en la última columna aparece el índice del arreglo correspondiente a cada proceso.

Comandos 2.14:

$ qstat -f

```
queuename                      qtype used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   0/2       0.03     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   1/1       0.00     lx26-x86
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50     1 3
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   1/2       0.00     lx26-x86
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50     1 2
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   0/2       0.05     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   0/2       0.97     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-5.local        BIP   0/2       0.00     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-6.local        BIP   1/2       0.00     lx26-x86
     35 0.55500 program    jzuluaga     r     11/02/2006 08:57:50     1 1
---------------------------------------------------------------------------------
all.q@compute-0-7.local        BIP   0/2       0.02     lx26-x86
---------------------------------------------------------------------------------
all.q@compute-0-8.local        BIP   0/2       0.04     lx26-x86
```

Una vez ejecutados los procesos los archivos con los standard output y el standard error aparecen como program.o35.1, program.o35.2, program.o35.3, program.e35.1, program.e35.2, program.e35.3, uno por cada proceso en el arreglo.

Es importante insistir que en el caso presentado antes, solamente el archivo input.dat distingue unos procesos de otros en el arreglo. Los archivos parameters.param y output.dat serán los mismos para todos los procesos dado que no existe, en este caso, ningún mecanismo interno en el programa que diferencie esos archivos en los procesos.

Una alternativa para la ejecución de múltiples procesos usando el mecanismo de arreglo de procesos se presenta a continuación:

Comandos 2.15:

$ mkdir run_{a,b,c}
$ cp program.out input_a.dat parameters.param run_a
$ cp program.out input_b.dat parameters.param run_b
$ cp program.out input_c.dat parameters.param run_c

El archivo único de lanzamiento sería:

launch-program-multiple-run.sh:

```
#!/bin/bash
#Launching script for program.out
#Submission options:
#$-S /bin/bash
#$-N program
#$-t 1-3:1

export PROGRAMTMP=/tmp
inputs=([1]=input_a.dat [2]=input_b.dat [3]=input_c.dat)
```

```
rundir=([1]=run_a [2]=run_b [3]=run_c)

cd ${rundir[$SGE_TASK_ID]}
./program.out ${inputs[$SGE_TASK_ID]}
```

2.3.Síntesis de comandos

Comando	Explicación
$ export PROGRAMTMP=/tmp	Asigna valor a la variable PROGRAMTMP
$ ./program.out input.dat	Lanza el programa program.out con entrada como input.dat
$ qsub launch-program.sh	Lanza el programa usando SGE
$ qacct -j 17	Examina las propiedades del trabajo 17
$ qstat	Monitorea el estado del proceso
$ qstat -f	Monitorea la lista de colas en el cluster
$ ls -l	Lista con detalles los archivos de un directorio
$ man qsub	Examina la página de manual de qsub
$ qsub -now y launch-program.sh	Lanza el programa con SGE solicitando al programador de trabajos que se ejecute inmediatamente
$ qdel 21	Detiene el proceso 21
$ mkdir run{1,2,3}	Crea 3 directorios run{1,2,3}

Segunda Parte
Guía Práctica del Administrador

Guía 3
Administración básica de la plataforma

Contenido sintético

3.Administración básica de la plataforma

3.1.Organización del sistema operativo

Rocks es una distribución basada en Redhat Enterprise Linux. La estructura de la distribución es por tanto similar en muchos aspectos a las distribuciones de ese mismo sabor (Fedora, CentOS, Scientific Linux, etc.) Las características especiales de trabajo en el cluster hacen sin embargo que hayan particularidades en la estructura del sistema operativo (servicios, sistemas de archivos, etc.) que sintetizaremos de forma práctica a continuación.

Sistemas de archivos locales

El disco del frontend y los nodos vienen particionado generalmente en 2 a 3 particiones: 1) La partición raíz (montada en el directorio '/') que contiene los archivos del sistema operativo y el espacio de almacenamiento de archivos temporales, logfiles, archivos de configuración, etc. 2) una partición especial para almacenamiento masivo en el frontend y en cada nodo que normalmente se monta sobre el directorio '/state/partition1'; en el frontend esta partición contiene las cuentas de usuario y otros archivos importantes relacionados con la instalación del sistema operativo; en los nodos esta partición puede usarse libremente para almacenar localmente grandes volúmenes de información. 3) en las últimas versiones de Rocks (>4.2) se ha incluido en el esquema de particionado por defecto una partición que se monta sobre el directorio '/var' que normalmente contiene información “variable” generada por los distintos programas y servicios del sistema operativo, incluyendo los logfiles.

NOTA:
Localmente en el frontend /state/partition1 tiene asociado el enlace simbólico /export:

Comandos 3.1:

```
# ls -ld /export
lrwxrwxrwx 1 root root 16 Nov 1 18:50 /export -> state/partition1
```

En lo sucesivo hablaremos directamente de /export en lugar de /state/partition1.

El esquema de particionado del frontend y los nodos configurado por defecto se muestra a continuación:

Comandos 3,2:

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             7.7G  3.7G  3.7G  51% /
none                  497M     0  497M   0% /dev/shm
/dev/hda5              63G  4.7G   55G   8% /state/partition1
/dev/hda2             3.9G  292M  3.4G   8% /var
```

Comandos 3,3:

```
# ssh c0-7 df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             7.7G  2.9G  4.5G  39% /
none                  502M     0  502M   0% /dev/shm
/dev/sda5             135G  819M  127G   1% /state/partition1
/dev/sda2             3.9G   74M  3.6G   2% /var
```

De los sistemas de archivos locales en el frontend vale la pena resaltar los siguientes directorios de gran relevancia para el cluster:

- /export/home (/state/partition1/home):

```
total 68
drwxr-xr-x 3 condor condor 4096 Nov 1 14:22 condor
drwxr-xr-x 7 root root 4096 Nov 1 14:24 install
drwx------ 7 fulano fulano 4096 Nov 8 11:34 fulano
```

  Este directorio contiene de un lado todos los home directory de los usuarios. De otra parte se encuentra allí también el directorio del usuario condor donde se depositan importantes archivos del sistema de colas de Condor. También encontramos allí el directorio install que describimos a continuación.

- /export/home/install:

```
total 20
drwxr-xr-x 3 root root 4096 Nov 1 14:24 contrib
drwxr-xr-x 4 root root 4096 Nov 1 14:27 rocks-dist
drwxr-xr-x 13 root root 4096 Nov 1 18:58 rolls
drwxr-xr-x 3 root root 4096 Nov 1 15:37 sbin
drwxr-xr-x 3 root root 4096 Nov 1 19:12 site-profiles
```

  Este importante directorio contiene la totalidad de los paquetes de instalación, archivos de configuración, programas y scripts especiales que usa Rocks para realizar la instalación del sistema operativo en los nodos.

  De los subdirectorios que encontramos allí son de resaltar los siguiente:
  - rolls: que contiene todos los paquetes instalados con los Rolls agregados al sistema desde la instalación o en un momento posterior a ella.

- site-profiles: donde se almacenan distintos archivos de configuración usados para personalizar la instalación de los nodos de computo en el cluster.
- rocks-dist: este subdirectorio, cuyo contenido es creado con el comando 'rocks-dist dist' contiene los archivos finales que serán instalados en los nodos del cluster. Al conjunto de estos archivos se lo conoce en Rocks como la “Distribución”.

Sistemas de archivos por red

Además de los sistemas de archivos locales (asociados a las particiones en el disco duro del frontend y de cada nodo) Rocks maneja un conjunto de sistemas de archivos que se montan por red usando el servicio NFS (Network File System) en los nodos. Estos sistemas de archivos permiten que contenidos críticos del frontend puedan ser accedidos localmente en cada nodo. El montaje de sistemas archivos por red usando NFS tiene un costo en el uso de la red que debe minimizarse en un entorno de computación distribuida. Por esa razón se usa adicionalmente un servicio que permite el montaje “automático” de solo aquellos directorios de los sistemas de archivos por red, que son requeridos por el usuario en los nodos.

La lista de los sistemas de archivos del frontend que pueden ser montados en los nodos puede usarse el comando exportfs:

Comandos 3,4:

```
# exportfs -v
/state/partition1
10.0.0.0/255.0.0.0(rw,async,wdelay,root_squash)
```

El contenido del directorio /export es:

Comandos 3,5:

```
# ls -l /export/
total 32
drwxr-xr-x 2 root root 4096 Nov 9 05:09 apps
drwxr-xr-x 19 root root 4096 Nov 8 15:16 home
drwxr-xr-x 3 root root 4096 Nov 1 19:12 site-roll
```

Para montar automáticamente los directorios contenidos en /export sobre los nodos se configura el sistema autofs a través de los archivos /etc/auto.master, /etc/auto.home, /etc/auto.share. Normalmente estos archivos deben residir en el directorio /etc de todas las máquinas del cluster incluyendo el propio frontend.

/etc/auto.master:

```
/share /etc/auto.share --timeout=1200
/home /etc/auto.home --timeout=1200
```

/etc/auto.home:

```
install cluster.local:/export/home/&
condor cluster.local:/export/home/condor
fulano cluster.local:/export/home/fulano
```

/etc/auto.share:

```
apps cluster.local:/export/&
install cluster.local:/export/home/&
```

El archivo auto.master contiene la lista de los directorios sobre los que se montan los sistemas de archivos por red descritos en detalle en los archivos de configuración respectivos. Así por ejemplo en cada máquina del cluster sobre el directorio /home se monta sistemas de archivos con las reglas descritas en el archivo de configuración /etc/auto.home. La opción '--timeout' permite definir un tiempo de espera después del cual, si el sistema de archivos no esta siendo utilizado por el usuario en la máquina remota entonces se desmonta automáticamente para liberar la sobrecarga de red que este procedimiento produce. De estos archivos el /etc/auto.home es probablemente el más importante. En el se definen, entre otras, las reglas para montar los home directory de los usuarios en los nodos. La sintaxis general de las entradas del archivo /etc/auto.* es:

<directorio-local> <servidor-nfs>:<sistema-de-archivos>/<directorio-remoto>

Donde el <directorio-local> es el nombre del directorio sobre el que se deberá montar el <directorio-remoto> que en la maquina <servidor-nfs> esta sobre el sistema de archivos <sistema-de-archivos>.

El resultado de esta configuración es que cuando el usuario se conecta a cualquier nodo del cluster su home directory /home/fulano es automáticamente montado vía NFS desde el directorio /export/home/fulano que se encuentra en el frontend. Una prueba de esto se puede verse conectándose a un nodo y ejecutando el comando:

Comandos 3,6:

```
# ssh c0-8
# df -ht nfs
Filesystem            Size  Used Avail Use% Mounted on
cluster.local:/export/home/fulano
                       63G  4.7G   55G   8% /home/fulano
```

De otra parte el archivo de configuración /etc/auto.share permite configurar, entre otro, el montaje automático del directorio /export/apps, donde se pueden instalar o almacenar archivos que se requiera sean accedidos desde todos los nodos del cluster. Según el archivo /etc/auto.master el directorio /export/apps es montado automáticamente en cada nodo en el directorio /share/apps. Este directorio es particularmente útil cuando se desea instalar aplicaciones, bibliotecas, bases de datos, etc. que se quiere acceder desde todos los nodos del cluster.

Tips útiles

Cuando se realizan cambios en los archivos de configuración del servicio autofs el servicio debe “recargarse”. Esto se realiza usando el comando 'service' de linux como se ilustra a continuación:

Comandos 3,7:

```
# service autofs reload
Checking for changes to /etc/auto.master ....
Reload map /usr/sbin/automount --timeout=1200 /share file /etc/auto.share
Reload map /usr/sbin/automount --timeout=1200 /home file /etc/auto.home
```

Se puede recargar el servicio también en otros (o todos) los nodos del cluster usando cluster-fork:

Comandos 3,8:

```
# cluster-fork service autofs reload
compute-0-0:
Checking for changes to /etc/auto.master ....
Reload map /usr/sbin/automount --timeout=1200 /share file /etc/auto.share
Reload map /usr/sbin/automount --timeout=1200 /home file /etc/auto.home
compute-0-1:
Checking for changes to /etc/auto.master ....
Reload map /usr/sbin/automount --timeout=1200 /share file /etc/auto.share
Reload map /usr/sbin/automount --timeout=1200 /home file /etc/auto.home
...
```

3.2.El Servicio 411

Como hemos visto hasta aquí el cluster es un sistema en el que información de todo tipo transita de forma más o menos transparente a través de todas las componentes de la plataforma: los sistemas de archivos del frontend pueden ser accedidos desde los nodos, los usuarios pueden conectarse con una sola cuenta de usuario en todos los nodos del cluster, no es necesario ingresar ninguna contraseña para acceder a los mismos entre otras cosas.

Algunas de estas cosas son posibles gracias al servicio ***411 Secure Information System***. Este servicio permite que archivos de configuración vitales para los servicios del cluster (listas de usuarios, tabla de passwords, grupos, configuración del servicio autofs, entre otras) sean compartidos por todas las máquinas de la plataforma, garantizando además (y ofreciendo las herramienta necesarias para) que se mantengan sincronizados a lo largo de la operación del cluster.

Para dar un ejemplo del papel de este servicio cada vez que se crea un usuario en el frontend se agrega una entrada a los archivos /etc/passwd, /etc/shadow, /etc/group y /etc/auto.home. 411 se encarga de que esos archivos se propaguen por todos los nodos del cluster de modo que el nuevo usuario pueda conectarse sin problemas a esos nodos.

Tips útiles

411 utiliza servicios web para publicar en el frontend (de manera encriptada, de ahí su nombre de seguro) los archivos de configuración que van a ser compartidos y para descargar en cada nodo esos mismos archivos.

Esto implica que para que el servicio funcione debe garantizarse que el servidor web del frontend este funcionando. Para verificarlo se puede usar el comando:

Comandos 3,9:

```
# service httpd status
```

Si el servicio esta activo la sálida sería:

```
httpd (pid 30585 30584 30583 30582 30581 30580 30579 30578 30575) is running...
```

De lo contrario se obtendría:

```
httpd is stopped
```

Si por alguna razón el servidor web no esta corriendo es necesario iniciarlo:

Comandos 3,10:

```
# service httpd start
Starting httpd: [ OK ]
```

La lista de los archivos compartidos usando 411 puede encontrarse en el archivo>

/ var/411/Files.mk:

```
...
AUTOMOUNT = #(wildcard /etc/auto.*)
# These files all take a "#" comment character.
# If you alter this list, you must do a 'make clean; make'.
FILES = #(AUTOMOUNT) \
/etc/passwd \
/etc/shadow \
/etc/group \
/etc/services \
/etc/rpc
# FILES += /my/file
...
```

El servicio 411 esta configurado para realizar en forma automática la sincronización de los archivos de configuración en el cluster. Sin embargo en algunas situaciones es necesario "forzar" la sincronización después de que se ha hecho un cambio en los archivos de configuración (creación de un usuario, modificación de la configuración del servicio autofs, creación de un nuevo grupo, etc). La sincronización se puede realizar de tres maneras diferentes:

1) Usando service:

Comandos 3,11:

```
# service 411 commit
Commiting changes to login files using 411.
411 Wrote: /etc/411.d/etc.shadow
Size: 2727/1840 bytes (encrypted/plain)
Alert: sent on channel 255.255.255.255:8649 with master 10.1.1.1
```

Este mecanismo solamente actualiza los archivos de configuración que han cambiado recientemente.

2) Usando make:

Comandos 3,12:

```
# make -C /var/411
```

Que es equivalente a 1) con la diferencia de que se puede forzar la sincronización de todos los archivos usando la regla “force”:

Comandos 3,13:
```
# make -C /var/411 force
```

3) Usando 411get en los nodos:

Comandos 3,14:
```
# cluster-fork 411get
compute-0-0:
/etc.rpc
/etc.group
/etc.shadow
/etc.auto..misc
/etc.auto..master
/etc.passwd
/etc.auto..home
/etc.auto..net
/etc.services
/etc.auto..share
...
```

Este mecanismo es particularmente útil para detectar problemas en la transferencia de los archivos entre el frontend y los nodos que pueden no detectarse con los mecanismos anteriores. En particular si el servidor web esta caído el comando anterior tendría como sálida:
```
compute-0-0:
Error: Could not reach a master server. Masters: [http://10.1.1.1/411.d/ (-1)]
compute-0-1:
Error: Could not reach a master server. Masters: [http://10.1.1.1/411.d/ (-1)]
```

Lo que se resolvería iniciando el servicio httpd como se explico más arriba.

3.3.Administración de usuarios

La administración de las cuentas de usuario y su configuración esta entre las tareas más importantes del trabajo del root en el cluster. Rocks normalmente incluye mecanismos automáticos para la administración de usuarios. No es sin embargo despreciable en términos prácticos conocer un poco a fondo los mecanismos detallados que se utilizan para crear, modificar y eliminar cuentas de usuario en el cluster:

Creación de cuentas de usuario

La creación de una cuenta de usuario en el cluster se realiza en 5 pasos:

1) Creación básica de la cuenta. Esto se logra mediante el uso del comando de linux useradd:

Comandos 3,15:
```
# useradd usuario
```

Este comando crea una entrada para el usuario en los archivos /etc/passwd, /etc/shadow, /etc/group (por defecto se crea un grupo por usuario). Adicionalmente crea el home directory /export/home/usuario y copia allí el contenido del directorio /etc/skel (archivos de configuración básicos). Una vez creada la cuenta el home directory contendrá,

Comandos 3,16:

```
# ls -al /export/home/usuario
total 28
drwx------ 2 usuario usuario 4096 Nov 9 07:30 .
drwxr-xr-x 20 root root 4096 Nov 9 07:30 ..
-rw-r--r-- 1 usuario usuario 24 Nov 9 07:30 .bash_logout
-rw-r--r-- 1 usuario usuario 191 Nov 9 07:30 .bash_profile
-rw-r--r-- 1 usuario usuario 124 Nov 9 07:30 .bashrc
-rw-r--r-- 1 usuario usuario 383 Nov 9 07:30 .emacs
-rw-r--r-- 1 usuario usuario 120 Nov 9 07:30 .gtkrc
```

Una vez cread la cuenta es necesario que asignemos correctamente la ubicación del home directory. Por defecto useradd fija el home directory en el /export/home. Sin embargo para que el usuario encuentre su home directory en otros nodos este debe ser cambiado a /home. Para hacerlo se debe usar el comando usermod:

Comandos 3,17:

```
# usermod -d /home/usuario usuario
```

2) Asignación de una contraseña. La contraseña se asigna normalmente usando el comando passwd:

Comandos 3,18:

```
# passwd usuario
```

Tips útiles

Es común en la administración del cluster que en un momento dado se necesite crear muchas cuentas simultáneamente. Si bien la creación misma con useradd no ofrece ninguna incomodidad la fijación de las contraseñas puede ser un trabajo extenuante. Se puede para ello recurrir sin embargo al comando chpasswd que permite fijar muchas contraseñas preestablecidas a un conjunto de usuarios con un solo comando.

A continuación se presenta un script que permite la creación de un conjunto de usuario a partir de una lista de logines almacenadas en un archivo de texto plano 'logins.txt':

manyuseradd.sh:

```
suffix=2006
for login in $(cat logins.txt)
do
echo "Creating $login..."
useradd $login
echo "$login:$login-$suffix" >> passwords.txt
done
chpasswd < passwords.txt
rm passwords.txt
```

Notese que los passwords se crearon usando el mismo login y un sufijo común. Es muy importante que los usuarios cambien sus contraseñas después de conectarse por primera vez a su cuenta para garantizar así la seguridad de la plataforma.

La creación de la cuenta no es sin embargo garantía de que el usuario pueda conectarse de manera transparente en el cluster, es necesario además hacer un commit de los archivos de configuración con 411 a todos los nodos del cluster y configurar el autofs para que el home directory sea montado automáticamente en los nodos.

3) Configuración del autofs. Para que el usuario pueda encontrar su home directory en cualquier nodo del cluster es necesario agregar al archivo de configuración /etc/auto.home una línea con la sintaxis que se presento en la sección 1 de este documento.

/etc/auto.home:

```
...
usuario cluster.local:/export/home/usuario
```

4) Sincronización de los archivos de configuración usando 411. Después de que se han completado los pasos anteriores, los archivos de configuración con la información definida en ellos deben sincronizarse en todo el cluster. Se recomienda usar el método make con "force" para garantizar que todos los archivos son sincronizados:

Comandos 3,19:

```
# make -C /var/411 force
```

5) Una vez sincronizados los archivos de configuración (incluyendo los del servicio autofs) es necesario recargar este servicio en todo el cluster para garantizar que no existan inconvenientes en el montaje de los home directory:

Comandos 3,20:

```
# service autofs reload
# cluster-fork service autofs reload
```

Una vez se han completado estos 5 pasos el usuario puede ahora hacer uso de todos los servicios del cluster. Es importante en este punto verificar que todo funcione correctamente. Sugerimos la siguiente lista de verificación:

Lista de chequeo para una cuenta de usuario

1. ¿El usuario puede conectarse al frontend?.

Comandos 3,21:

```
# ssh usuario@localhost
```

Si la respuesta es negativa:

- El password puede tener problemas. Asignelo nuevamente
- Verifique que la cuenta aparezca en el archivo /etc/passwd

2. ¿El usuario encuentra su home directory?.

Comandos 3,22:

```
# pwd
/home/usuario
```

Si la respuesta es negativa:
- Puede ser necesario reiniciar el servicio autofs.

3. ¿Una vez conectado al frontend puede el usuario conectarse a otros nodos sin necesidad de una contraseña?

Comandos 3,23:

```
# ssh c0-0 w
```

Si la respuesta es negativa, es decir el nodo pide al usuario una contraseña:
- Puede ser necesario reiniciar el servicio autofs en el nodo (cluster-fork service autofs reload)
- Verifique que en el archivo /etc/passwd el directorio casa del usuario este como /home/usuario y no como /export/home/usuario
- Sincronice todos los archivos de configuración

Tips útiles

En las versiones de Rocks posteriores a la versión 4.2 los pasos anteriores pueden resumirse usando el comando rocks-user-sync, solamente a 3 pasos:

1. Creación de la cuenta con useradd.
2. Fijación de la contraseña con passwd
3. Ejecución del comando:

Comandos 3,24:

```
# rocks-user-sync
```

rocks-user-sync realiza automáticamente los pasos 3, 4 y 5 del procedimiento completo descrito más arriba.

Una vez creada la cuenta con este mecanismo siempre será apropiado verificar que la cuenta de usuario funcione correctamente usando la lista de chequeo del apartado anterior.

Modificación de cuentas de usuario

Una vez creada las propiedades de una cuenta de usuario (exceptuando su nombre o login) pueden ser modificadas. Las propiedades más comúnmente sujetas a modificación incluyen la contraseña y el grupo o grupos a los que pertenece la cuenta.

Para modificar por ejemplo el grupo al que pertenece un usuario se utiliza el comando usermod:

Comandos 3,25:

```
# usermod -g <gid> usuario
```

Donde <gid> es el identificador del grupo al que se desea pertenezca el usuario.

Una vez modificada alguna de las propiedades de la cuenta deben sincronizarse nuevamente los archivos de configuración en el cluster. En este caso por simplicidad puede usarse el comando service para hacerlo:

Comandos 3,26:

```
# service 411 commit
```

Tenga en cuenta que de no hacer la sincronización manual, 411 realiza una sincronización automática cada hora.

Eliminación de cuentas de usuario

La eliminación de una cuenta de usuario se realiza en 3 pasos:

1) Eliminación de la cuenta. Para ello se usa el comando userdel:

Comandos 3,27:

```
# userdel usuario
```

2) Desmontado del home directory (en caso de que una sesión del usuario haya sido abierta hace poco):

Comandos 3,28:

```
# umount /home/usuario
# cluster-fork umount /home/usuario
```

3) Eliminación del home directory:

Comandos 3,29:

```
# rm -rf /export/home/usuario
```

4) Sincronización de los archivos

El borrado de la cuenta deja la línea de configuración del autofs en el archivo /etc/auto.home Aunque esta línea no hace ningún daño puede eliminarse para que no introduzca ruido en este archivo.

3.4.Síntesis de comandos

Comando	Explicación
# ls -ld /export	Muestra las propiedades del enlace simbólico /export
# exportfs -v	Muestra la lista de los sistemas de archivos compartidos vía NFS
# ls -l /export/	Revisa el contenido del directorio /export
# df -ht nfs	Muestra los sistemas de archivos tipo nfs montados en la máquina
# service autofs reload	Recarga el servicio autofs
# cluster-fork service autofs reload	Recarga el servicio autofs en todo el cluster
# service httpd status	Muestra el estado del servicio httpd
# service httpd start	Inicia el servicio httpd
# service 411 commit	Sincroniza los archivos de configuración que han cambiado recientemente
# make -C /var/411	Igual que el anterior
# make -C /var/411 force	Sincroniza TODOS los archivos de configuración
# cluster-fork 411get	Obtiene los archivos de configuración del frontend (operación en todos los nodos)
# useradd usuario	Crea una cuenta de usuario
# ls -al /export/home/usuario	Muestra el contenido (incluso el oculto) del directorio casa de usuario
# usermod -d /home/usuario usuario	Cambia el directorio casa del usuario en los archivos de configuración
# passwd usuario	Fija la contraseña de usuario
# ssh usuario@localhost	El root se conecta como usuario al frontend
# rocks-user-sync	Finaliza la creación de cuentas de usuario (se ejecuta después de useradd y passwd)
# usermod -g <gid> usuario	Cambia el número de grupo de un usuario
# userdel usuario	Borra la cuenta de usuario
# umount /home/usuario	Desmonta el home directory de usuario
# cluster-fork umount /home/usuario	Desmonta el home directory de usuario en todo el cluster
# rm -rf /export/home/usuario	Borra el home directory en el frontend

Segunda Parte
Guía Práctica del Administrador

Guía 4
Tópicos especiales de administración

Contenido sintético

4.Tópicos especiales de administración

4.1.Instalación de nuevo software

La instalación de nuevo software en el frontend cluster se realiza siguiendo en principio procedimientos similares a los que se requieren para instalar software en cualquier servidor Linux. Sin embargo a la hora de requerir que el software pueda accederse desde todos los nodos, para ejecutarlo por ejemplo usando un Scheduler o para que las instancias de un programa en paralelo encuentren las componentes fundamentales del programa (bibliotecas, archivos de configuración, repositorios de temporales, etc.) es necesario configurar el paquete y los sistemas de archivos de manera apropiada.

Se describe a continuación la instalación de nuevo software en el cluster a través de distintos mecanismos: 1) instalación de un rpm de binarios, 2) instalación de un tarball de fuentes, 3) instalación de una biblioteca de rutinas. En cada caso se ilustra el proceso de instalación, configuración y puesta en funcionamiento de algunos paquetes que pueden ser de utilidad para el trabajo científico.

NOTA: El número de condiciones posibles para la instalación de paquetes en Linux es tan amplio que la guía ofrecida aquí solo debe considerarse como una referencia práctica a algunas situaciones comunes que se enfrentan al instalar software en el cluster. Inclusive, las soluciones o tips ofrecidos aquí pueden tener alternativas más eficientes y prácticas. En la realidad la instalación de un paquete en el cluster es un proceso casi único dependiente del paquete y de la experiencia, habilidad y conocimientos del administrador.

Instalación de un RPM con binarios

La instalación de binarios es quizá la opción más cómoda para la instalación de nuevo software en Linux. Sin embargo puede no ser necesariamente la mejor. De una parte el RPM tiene preconfigurado los lugares donde se instalaran los archivos que vienen con el paquete, lo que para una instalación en una plataforma distribuida puede no ser la mejor opción. En segunda instancia es común que la instalación de binarios este sometido a

dependencias de otros binarios lo que puede hacer muy incomodo el proceso de instalación.

En algunos casos prácticos sin embargo esta puede ser una buena opción para la instalación de paquetes simples o paquetes que no ofrecen grandes dificultades en la instalación.

Para ilustrar la instalación de binarios desde un RPM usaremos el **paquete gnuplot**, para graficación científica.

Antes de instalar un paquete con un RPM se debe verificar que no haya sido instalado previamente o que no venga con el sistema operativo:

Comandos 4,1:

```
# rpm -q gnuplot
package gnuplot is not installed
```

Cuando el paquete ya esta instalado aparecerá la información completa del paquete disponible, su versión y a veces información sobre la plataforma específica o distribución para el que fue preparado. En el caso específico de gnuplot se trata de un paquete que no viene incluido en la distribución (al menos en versiones anteriores a la 4.2.1).

Podemos verificar si el paquete esta instalado en otras máquinas (obviamente sin éxito por qué de no estarlo en el frontend es improbable que este instalado en los nodos del cluster) usando el comando cluster-fork:

Comandos 4,2:

```
# cluster-fork rpm -q gnuplot

package gnuplot is not installed
compute-0-1:
package gnuplot is not installed
compute-0-2:
package gnuplot is not installed
...
```

La instalación procede de la manera convencional usando el comando rpm:

Comandos 4,3:

```
# rpm -Uvh gnuplot-3.7.3-2.i386.rpm

Preparing...                ########################################### [100%]
   1:gnuplot                ########################################### [100%]
```

En este caso se comprueba afortunadamente que no existe ningún problema de dependencias y el paquete puede ser instalado sin inconvenientes.

Para instalar el paquete en todos los nodos del cluster podría procederse de dos formas distintas:

1) **Instalación en caliente**: con este procedimiento el paquete se instala directamente en los nodos después de que el sistema ha sido instalado completamente. Para hacerlo

primero debemos garantizar que desde cada nodo podamos tener acceso al archivo rpm para realizar la instalación. Se recomienda la creación de un directorio 'src' en /export/apps que como vimos en el Documento 1 es montado automáticamente usando autofs en todos los nodos del cluster:

Comandos 4,4:

```
# mkdir /export/apps/src
```

Una vez creado el repositorio de paquetes se puede verificar que el directorio pueda ser accedido desde los nodos a través del punto de montaje definido en el archivo auto.share:

Comandos 4,5:

```
# cluster-fork ls -l /share/apps
```

```
compute-0-1:
total 4
drwxr-xr-x  2 root root 4096 Nov 14 06:15 src
compute-0-2:
total 4
drwxr-xr-x  2 root root 4096 Nov 14 06:15 src
...
```

Una vez creado el repositorio copiamos allí el rpm que deseamos instalar:

Comandos 4,6:

```
# cp -rf gnuplot-3.7.3-2.i386.rpm /export/apps/src
```

Con el archivo rpm en su lugar podemos proceder ahora con la instalación en todos los nodos ejecutando con cluster-fork el comando rpm de instalación:

Comandos 4,7:

```
# cluster-fork rpm -Uvh /share/apps/src/gnuplot-3.7.3-2.i386.rpm
```

```
compute-0-0:
Preparing...                ##################################################
gnuplot                     ##################################################
compute-0-1:
Preparing...                ##################################################
gnuplot                     ##################################################
compute-0-2:
Preparing...                ##################################################
gnuplot                     ##################################################
...
```

Una vez realizada la instalación podemos verificar por ejemplo que el binario del programa este disponible, por ejemplo usando el comando which:

Comandos 4,8:

```
# cluster-fork which gnuplot
compute-0-0:
/usr/bin/gnuplot
compute-0-1:
```

```
/usr/bin/gnuplot
...
```

2) **Instalación desde la distribución**: Mas apropiado que instalar en caliente el paquete, es incluir el paquete directamente en la distribución que se instala en cada uno de los nodos. La ventaja evidente de este procedimiento estriba en el hecho que después de una re instalación de los nodos estará garantizado que el paquete se instale automáticamente sin requerir que se ejecuten las tareas descritas en el apartado anterior.

   Para incluir un paquete en la distribución que prepara rocks para los nodos se debe seguir el procedimiento descrito a continuación.

   Rocks tiene un espacio especialmente dedicado a las contibuciones adicionales de los usuarios a la distribución instalada. El espacio esta habilitado en el directorio /export/home/install/contrib/<versión>/<arch>/:

Comandos 4,9:

```
# ls -l /export/home/install/contrib/4.2.1/i386
total 8
drwxr-xr-x  2 root root 4096 Nov  1 14:24 RPMS
drwxr-xr-x  2 root root 4096 Nov  1 14:24 SRPMS
```

Allí se pueden colocar los RPMS que se desea agregar a la distribución. En nuestro caso por ejemplo:

Comandos 4,10:

```
# cp -rf gnuplot-3.7.3-2.i386.rpm /export/home/install/contrib/4.2.1/i386/RPMS
```

Copiar el archivo rpm allí no es suficiente. Ahora es necesario configurar la distribución para que incluya el nuevo paquete y después reconstruir una nueva versión de la misma.

La inclusión del paquete se hace a través de un archivo de configuración xml que debe prepararse en el directorio /export/home/install/site-profiles/<versión>/nodes:

Comandos 4,11:

```
# ls -l /export/home/install/site-profiles/4.2.1/nodes
total 4
-rw-rw-r--  1 root root 1964 Sep 25 00:29 skeleton.xml
```

El archivo skeleton.xml ofrece una plantilla para la preparación del nuevo archivo de configuración que deberá llamarse 'extend-compute.xml'. Para preparar este archivo debemos preparar una copia de skeleton.xml:

Comandos 4,12:

```
# cd /export/home/install/site-profiles/4.2.1/nodes
# cp skeleton.xml extend-compute.xml
```

Y editar este último archivo:

extend-compute.xml:

```
...
<changelog>
</changelog>

<main>
        <!-- kickstart 'main' commands go here, e.g., partitioning info -->
</main>

<!-- There may be as many packages as needed here. Just make sure you only
     uncomment as many package lines as you need. Any empty <package></package>
     tags are going to confuse rocks and kill the installation procedure
-->
<!-- <package> insert your 1st package name here and uncomment the line</package> -->
<!-- <package> insert your 2nd package name here and uncomment the line</package> -->
<!-- <package> insert your 3rd package name here and uncomment the line</package> -->
...
```

Para incluir el nuevo paquete (o nuevos paquetes) se deberá agregar en la sección “main” del xml una línea con la etiqueta <package> indicando SOLAMENTE el nombre del paquete (sin el número de la versión u otra información que venga con el archivo rpm). Después de editarlo en nuestro ejemplo el archivo extend-compute.xml deberá lucir así:

extend-compute.xml:

```
...
<changelog>
</changelog>

<main>
        <!-- kickstart 'main' commands go here, e.g., partitioning info -->
</main>

<package>gnuplot</package>
...
```

De existir más de un rpm se agregará una línea por cada rpm que se desea instalar.
Una vez configurado se debe reconstruir la distribución usando el comando de rocks, rocks-dist:

Comandos 4,13:

```
# rocks-dist dist
Cleaning distribution
Resolving versions (base files)
    including "kernel" (4.2.1,i386) roll...
    including "area51" (4.2.1,i386) roll...
    including "java" (4.2.1,i386) roll...
    including "condor" (4.2.1,i386) roll...
    including "web-server" (4.2.1,i386) roll...
    including "base" (4.2.1,i386) roll...
    including "grid" (4.2.1,i386) roll...
    including "sge" (4.2.1,i386) roll...
    including "hpc" (4.2.1,i386) roll...
    including "ganglia" (4.2.1,i386) roll...
    including "os" (4.2.1,i386) roll...
Including critical RPMS
...
```

El comando construye la estructura completa del directorio /export/home/install/rocks-dist. La estructura de ese directorio incluye enlaces simbólicos a los paquetes que serán instalados en los nodos. Estos enlaces simbólicos se encuentran en el subdirectorio lan/i386/RedHat/RPMS de rocks-dist. Es una buena idea verificar que se encuentren allí enlaces simbólicos a los nuevos paquetes:

Comandos 4.14:

```
# ls -ld rocks-dist/lan/i386/RedHat/RPMS/gnuplot-3.7.3-2.i386.rpm
lrwxrwxrwx  1 root root 62 Nov 14 06:40 rocks-dist/lan/i386/RedHat/RPMS/gnuplot-3.7.3-2.i386.rpm ->
/home/install/contrib/4.2.1/i386/RPMS/gnuplot-3.7.3-2.i386.rpm
```

Una vez preparada la instalación se garantiza que en lo sucesivo los nodos contendrán los paquetes adicionales seleccionados.

Tips útiles

Es importante que una vez se reconstruye la distribución se pruebe al menos con la reinstalación de uno de los nodos que la distribución funciona correctamente.

Para reinstalar fácilmente un nodo del cluster se puede recurrir a mecanismos de automatización que vienen instalados con Rocks y que usan el sistema kickstart. La re instalación procede de la siguiente manera:

a) Se elimina del nodo respectivo el archivo /.rocks-release

Comandos 4.15:

```
# ssh c0-6 rm -rf /.rocks-release
```

Esto habilita una opción en el gestor de arranque que hace que el nodo se reinicie la próxima vez en modo de re instalación.

b) Se inicia el proceso de re instalación:

Comandos 4.16:

```
# ssh c0-6 /boot/kickstart/cluster-kickstart
Shutting down kernel logger: [  OK  ]
Shutting down system logger: [  OK  ]
```

Ahora solo hace falta esperar. Es mejor si se tiene acceso a la consola del nodo para verificar si se producen errores fatales durante la re instalación (lo que lamentablemente no es difícil debido a la relativa fragilidad de los archivos de configuración xml que usa rocks para configurar la distribución). En caso de producirse errores deberá revisarse el archivo extend-compute.xml y reconstruir nuevamente la distribución en caso de cambios importantes a este archivo.

Instalación de un tarball con fuentes

La instalación de un tarball con fuentes es probablemente la manera más sana de instalar un nuevo paquete en el cluster. Si bien es un poco más incomoda y no goza del beneficio INMEDIATO de poder incluirse en una instalación fresca de los nodos, permite configurar en detalle los aspectos completos de la preparación de los binarios y de los directorios a los que irán a parar los archivos de instalación.

Para ilustrar la instalación de binarios desde un RPM usaremos el **paquete de rendering 3D**

povray (http://www.povray.org).

Antes de instalar paquetes en el cluster recomiendo crear en el directorio /export/apps una estructura de directorios apropiada que permita recibir los archivos que normalmente son instalados desde un tarball:

Comandos 4,17:

```
# cd /export/apps
# mkdir -p bin etc man sbin share src lib include
```

La instalación de un tarball se realiza en 5 pasos más o menos bien conocidos:

1) En primer lugar copiemos el tarball respectivo en el directorio desde el que realizaremos el proceso de instalación:

Comandos 4,18:

```
# cp povray-3.6.tar.gz /share/apps/src
```

2) Descomprimimos y desempacamos el tarball:

Comandos 4,19:

```
# cd /export/apps/src
# tar zxvf  povray-3.6.tar.gz
```

3) Configuramos el instalador:

Comandos 4,20:

```
# cd povray-3.6.1
# ./configure –prefix=/share/apps  --COMPILED_BY="Jorge Zuluaga <zuluagajorge@gmail.com>"
```

Debe tenerse especial cuidado en la inclusión de la opción --prefix=/share/apps que es la que garantizará que el programa quede instalado en una ubicación que pueda ser accedida desde todos los nodos del cluster. La opción COMPILED_BY es mandatoria.

Tips útiles

El script de configuración 'configure' normalmente tiene una larga lista de opciones que modifican el modo en el que se prepara la instalación. Las opciones pueden consultarse ejecutando

Comandos 4,21:

```
# ./configure --help
```

4) Compilamos las fuentes:

Comandos 4,22:

```
# make
```

NOTA: se puede con povray reemplazar este comando por 'make check' para que al terminar el instalador pruebe a hacer un test de rendering.

5) Instalamos el programa (si todo ha salido bien con la compilación en el paso anterior):

Comandos 4,23:

```
# make install
```

6) El paso final consiste en la configuración del sistema (en caso de ser necesario) para que los binarios del nuevo programa puedan ser encontrados sin problemas en el PATH de los usuarios. En el frontend esto se puede hacer cambiando el archivo /etc/profile:

/etc/profile:

```
...
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
...
```

Para que por ejemplo el directorio /share/apps/bin sea incluido en el path por defecto de los usuarios (incluido el administrador) basta agregar antes de la línea con el 'export' la línea:

PATH=$PATH:/share/apps/bin

Una vez hecho esto se debe garantizar que la misma línea se agregue a los archivos /etc/profile de los nodos del cluster. Utilizando lo aprendido en el documento 1 esto puede hacerse automáticamente usando el sistema 411. Para ello bastaría agregar a la lista de archivos en /var/411/Files.mk el archivo /etc/profile:

/var/411/Files.mk:

```
...
FILES = $(AUTOMOUNT) \
        /etc/passwd \
        /etc/shadow \
        /etc/group \
        /etc/services \
        /etc/rpc \
        /etc/profile
...
```

Regenerar el repositorio de archivos de configuración:

Comandos 4,24:

```
# make -C /var/411 clean
# make -C /var/411
```

Y enviar a los nodos los nuevos archivos de configuración:

Comandos 4,25:

```
# cluster-fork 411get
```

```
c0-0:
/etc.rpc
/etc.group
/etc.shadow
/etc.auto..misc
/etc.auto..master
/etc.passwd
/etc.profile
/etc.auto..home
/etc.auto..net
/etc.services
/etc.auto..share
...
```

Tips útiles

Para probar el funcionamiento de povray se puede intentar hacer el rendering de una de las escenas de ejemplo que vienen con el paquete. Para ello siga los siguientes pasos:

1) Loggeese como un usuario

2) Copie en el directorio casa del usuario uno de los escenarios de ejemplo

Comandos 4,26:

```
$ cp /share/apps/src/povray-3.6.1/scenes/advanced/sunsethf.pov .
```

3) Ejecute povray en el frontend:

Comandos 4,27:

```
$ povray +D +W640 +H480 +I sunsethf.pov
```

4) Ejecute povray en uno de los nodos:

Comandos 4,28:

```
$ ssh c0-5 /share/apps/bin/povray +D +W640 +H480 +I sunsethf.pov
```

Notese que en el último caso es necesario indicar el path completo del binario porque en este tipo de sesiones de ssh las variables del archivo profile no se activan.

El resultado de los comandos anteriores debe producir adicionalmente una imágen 'sunsethf.png' que puede visualizarse usando el comando:

Comandos 4,29:

```
$ display sunsethf.png
```

La imágen generada se muestra a continuación:

Figura 1. Ejemplo de imágen producida con povray

Instalación de una biblioteca

La instalación de bibliotecas en un cluster es un capítulo muy especial de la instalación de software en este tipo de plataformas. Su instalación no difiere mucho de la instalación de paquetes convencionales como se describió más arriba, exceptuando tareas de configuración específicas que deben realizarse para que las bibliotecas de enlace dinámico puedan ser correctamente enlazadas en tiempo de ejecución desde todos los nodos.

Ilustraremos estos conceptos con la librería para cálculo científico en C **Gnu Scientific Library (GSL)** (http://www.gnu.org/software/gsl).

Para instalar la biblioteca copiamos el tarball con los sources en el directorio compartido /export/apps/src donde hemos colocado los instaladores de las otras aplicaciones:

Comandos 4,30:

```
# cp -rf gsl-1.6.tar.gz /export/apps/src
```

La instalación procede de manera casi idéntica a la instalación de un paquete como povray.

1) Descomprimimos y desempacamos las fuentes,

Comandos 4,31:

```
# cd /export/apps/src
# tar zxvf gsl-1.6.tar.gz
```

2) Configuramos el instalador:

Comandos 4,32:

```
# cd gsl-1.6
# ./configure --prefix=/share/apps
```

3) Compilamos las fuentes:

Comandos 4,33:

```
# make
```

4) Instalamos la librería:

Comandos 4,34:

```
# make install
```

NOTA: puede ser una buena idea una vez instalada la librería limpiar de binarios el directorio de fuentes en tanto estos ya han sido copiados a los directorios respectivos del sistema. Para limpiar el directorio de fuentes de los binarios se puede usar:

Comandos 4,35:

```
# make clean
```

Para verificar si la instalación fue exitosa se pueden buscar los binarios de la librería en el directorio /share/apps/lib:

Comandos 4,36:

```
# ls /share/apps/lib
libgsl.a        libgslcblas.so        libgsl.la    libgsl.so.0.7.0
libgslcblas.a   libgslcblas.so.0      libgsl.so    pkgconfig
libgslcblas.la  libgslcblas.so.0.0.0  libgsl.so.0
```

Allí se encontraran las versiones de enlazado estático (.a, .la) y dinámico de la librería (.so.*).

Para verificar su funcionamiento se puede escribir un sencillo programa que haga uso de algunas de las rutinas de la librería:

testgsl.c:

```
#include <gsl/gsl_sf_trig.h>

int main(void)
{
  double x,dx;
  gsl_sf_result y;

  x=1;
  dx=0.1;

  gsl_sf_sin_err_e(x,dx,&y);

  printf("sin(%lf+/-%lf) = %lf+/-%lf\n",x,dx,y.val,y.err);
}
```

La compilación de un programa que hace uso de rutinas de una librería, como se sabe, no

es trivial, dado que es necesario indicarle al compilador la ubicación de los archivos de cabecera y de la librería misma para que haga la compilación y el enlazado:

Comandos 4,37:

```
$ gcc -I/share/apps/include -c testgsl.c -o testgsl.o
$ gcc testgsl.o -L/share/apps/lib -lgslcblas -lgsl -o testgsl.out
```

Esto puede abreviarse si se escribe un Makefile para la compilación. Notese que se esta usando en este caso una cuenta de usuario para probar el uso de la librería lo que nos permitirá más abajo hacer una prueba sobre todo el cluster aprovechando el montaje del home directory en los nodos (algo que no sería posible si trabajáramos directamente en el home directory del root). El resultado de este proceso es un ejecutable que ahora podemos intentar poner a funcionar:

Para evitar el uso de las opciones -I y -L se pueden cambiar las variables de ambiente C_INCLUDE_PATH y LIBRARY_PATH que se pueden agregar a los archivos /etc/profile y /etc/bashrc

Comandos 4,38:

```
$ ./testgsl.out
./testgsl.out: error while loading shared libraries: libgslcblas.so.0: cannot open shared object file: No such file or directory
```

Este error que es común cuando se usan librerías compartidas (share libraries) en Linux constituye la fuente de los aspectos que hacen de la instalación de una librería en el cluster un capítulo aparte de la instalación de otro tipo de software.

La solución más sencilla a este problema consiste en la inicialización de la variable de ambiente LD_LIBRARY_PATH que utiliza el “enlazador” al momento de la ejecución del binario para buscar los módulos compartidos que requiere el programa:

Comandos 4,39:

```
$ export LD_LIBRARY_PATH=/share/apps/lib
$ ./testgsl.out
sin(1.000000+/-0.100000) = 0.841471+/-0.054030
```

Esta solución permitirá la ejecución de binarios enlazados a gsl solo durante la sesión de shell actual. Si se desea por ejemplo ejecutar el programa en un nodo remoto se obtendrá el mismo mensaje de error de más arriba:

Comandos 4,4:

```
$ export LD_LIBRARY_PATH=/share/apps/lib
$ ssh c0-8 ./testgsl.out
./testgsl.out: error while loading shared libraries: libgslcblas.so.0: cannot open shared object file: No such file or directory
```

Si se desea hacerla funcionar en todas las sesiones deberá incluirse la línea del export en el archivo de configuración inicial '.bashrc' del usuario.

La solución más elegante y general al problema sin embargo la tiene el administrador. Para

que el directorio /share/apps/lib sea incluido en los directorios que POR DEFECTO usa el "enlazador" para buscar las librerías se deberá agregar al archivo /etc/ld.so.conf la siguiente línea:

/etc/ld.so.conf:

```
include ld.so.conf.d/*.conf
/usr/ofed/lib
...
/share/apps/lib
```

Una vez agregada la línea se deberá refrescar el enlazador dinámico con el comando:

Comandos 4,41:

```
# ldconfig
```

Hecho esto ahora la ejecución del programa en el frontend se realizará sin problemas sin necesidad de definir la variable LD_LIBRARY_PATH. Sin embargo la ejecución en los nodos seguirá presentando el mismo inconveniente. La solución propuesta aquí para este problema es la de sincronizar el archivo de configuración ld.so.conf en todo el cluster usando el servicio 411. Para eso agregamos al archivo /var/411/Files.mk una línea por el archivo respectivo, se refresca la base de archivos de configuración de 411 y se sincronizan los archivos de configuración en el cluster.

Una vez hecho esto se debe finalmente refrescar el enlazador de los nodos usando:

Comandos 4,42:

```
# cluster-fork ldconfig
```

Y de ese modo la biblioteca quedará funcionando sin inconveniente en todo el cluster.

Comandos 4,43:

```
$ cluster-fork ./testgsl.out
compute-0-0:
sin(1.000000+/-0.100000) = 0.841471+/-0.054030
compute-0-1:
sin(1.000000+/-0.100000) = 0.841471+/-0.054030
...
```

1. Monitoreo de recursos por la web

Las tareas de monitoreo de los recursos del cluster que usamos en el documento 1 y muchas otras más pueden realizarse usando la interfaz web de Ganglia una poderosa y completa herramienta que viene instalada casi por defecto con todas las distribuciones de Rocks.

Para acceder a Ganglia se debe acceder a través de un browser a la página que por defecto Rocks instala en el frontend:

Comandos 4,44:

firefox http://cluster.local

La página principal de la página luce como se muestra en la figura .. El link Cluster Status nos llevará directamente a la ventana con las herramientas de monitoreo del cluster. La página principal de Ganglia se muestra en la figura ..

Tips útiles

Por razones de seguridad la página del cluster solo puede accederse desde el cluster mismo. Sin embargo es común que el administrador y los usuarios deseen tener acceso desde ubicaciones remotas. Para habilitar el acceso desde afuera del cluster se debe cambiar la configuración del firewall en el frontend.

Para hacerlo basta cambiar una entrada en el archivo /etc/sysconfig/iptables donde se consignan las reglas y restricciones básicas que tiene definido el firewall. El contenido por defecto del archivo es:

/etc/sysconfig/iptables:

```
...
# Allow these ports
-A INPUT -m state --state NEW -p tcp --dport ssh -j ACCEPT
# Uncomment the lines below to activate web access to the cluster.
#-A INPUT -m state --state NEW -p tcp --dport https -j ACCEPT
#-A INPUT -m state --state NEW -p tcp --dport www -j ACCEPT
...
```

Para premitir el acceso al webserver desde el exterior se debe decomentar la línea relacionada con esta regla:

/etc/sysconfig/iptables:

```
...
-A INPUT -m state --state NEW -p tcp --dport www -j ACCEPT
...
```

Una vez hecho esto se debe reiniciar el demonio del firewall:

Comandos 4.45:

service iptables restart

Figura 3. Página principal del cluster

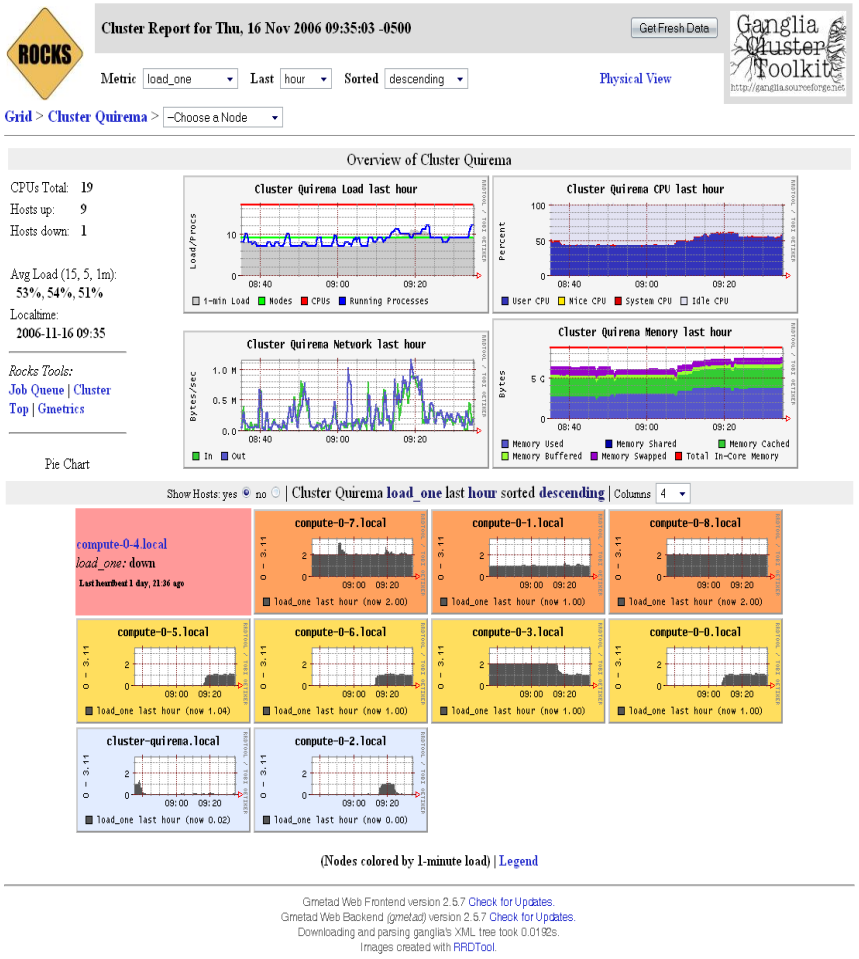


Figura 4. Página principal de Ganglia

Usando Ganglia se pueden monitorear, entre muchas otras cosas, los siguientes aspectos críticos del cluster:

- **Carga total de la plataforma**. En la página principal de Ganglia se reporta la carga total de la plataforma, incluyendo la carga de todos los procesadores (Load/Procs), el uso del ancho de banda en la red (Bytes/sec) y el uso de la memoria total disponible en el cluster.

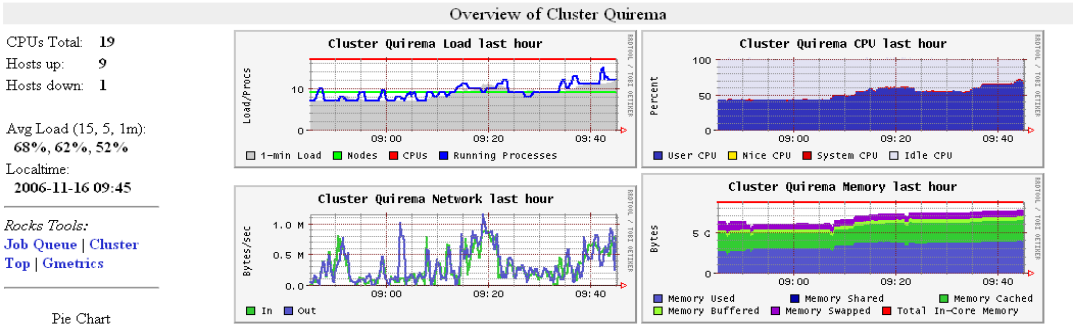


Figura 5. Parámetros globales del cluster

- **Carga de cada nodo del cluster**. Un historial del uso de los nodos codificado con colores que miden el estado de ocupación de cada uno es presentado también en la página principal de Ganglia.

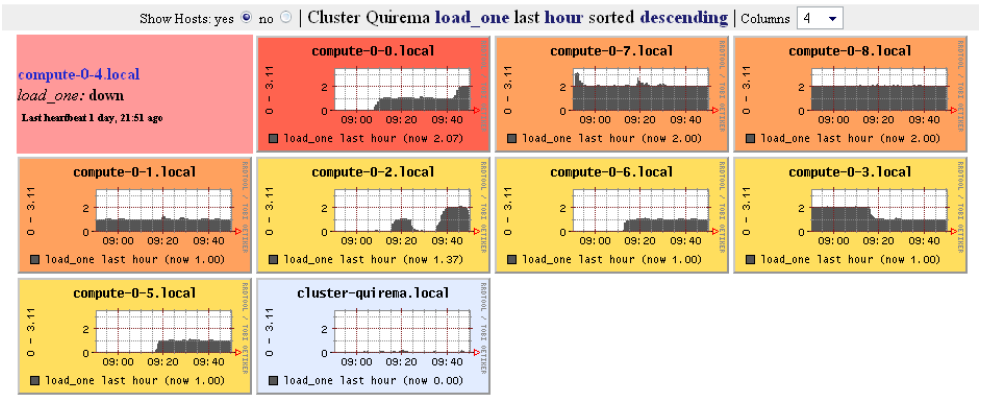


Figura 6. Carga de los nodos

A través de la caja de color de cada nodo es posible acceder a información estática y dinámica detallada de ese nodo.

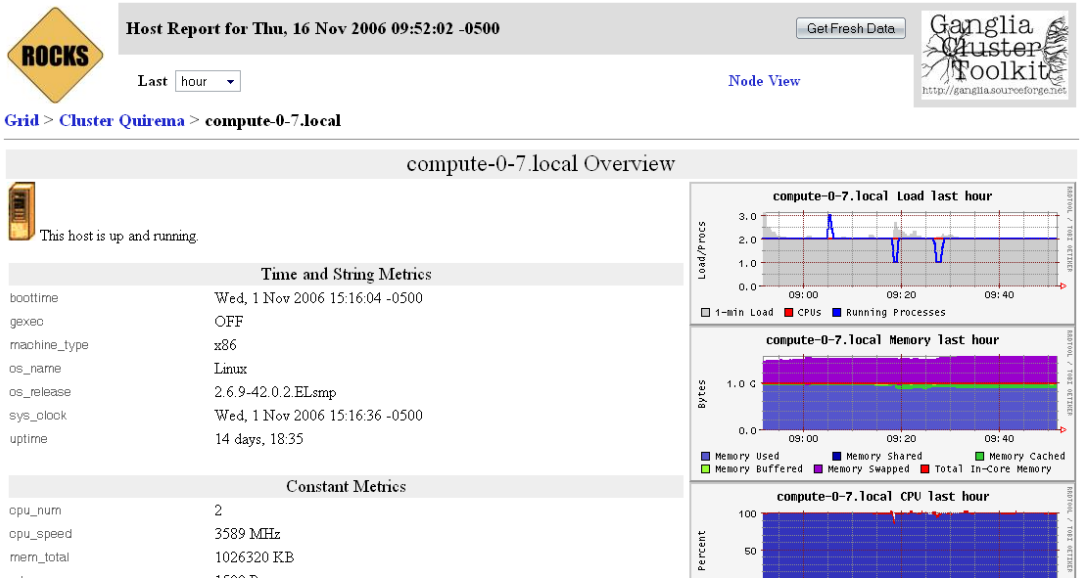
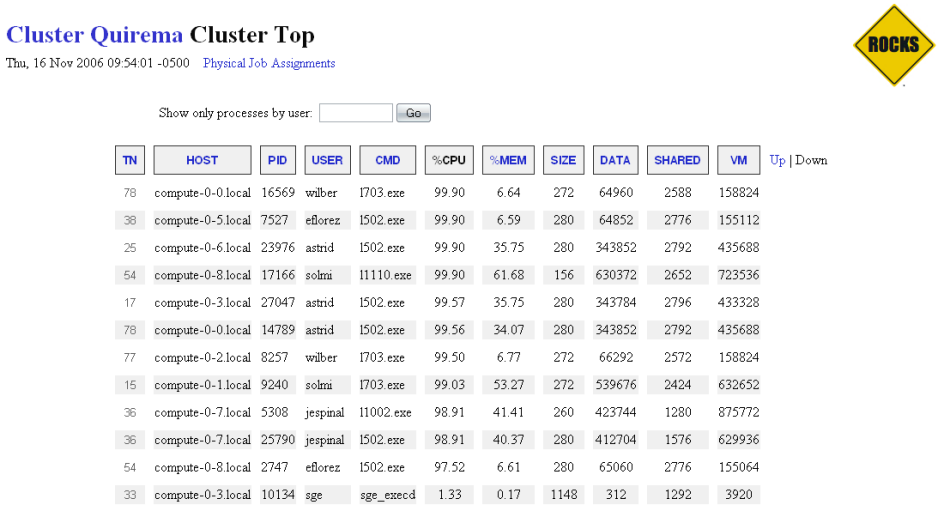


Figura 7. Información detallada de cada nodo

- **Trabajos en ejecución**. Es posible a través de Ganglia conocer también la lista de los procesos que se están ejecutando en los nodos.



Cluster Quirema Cluster Top

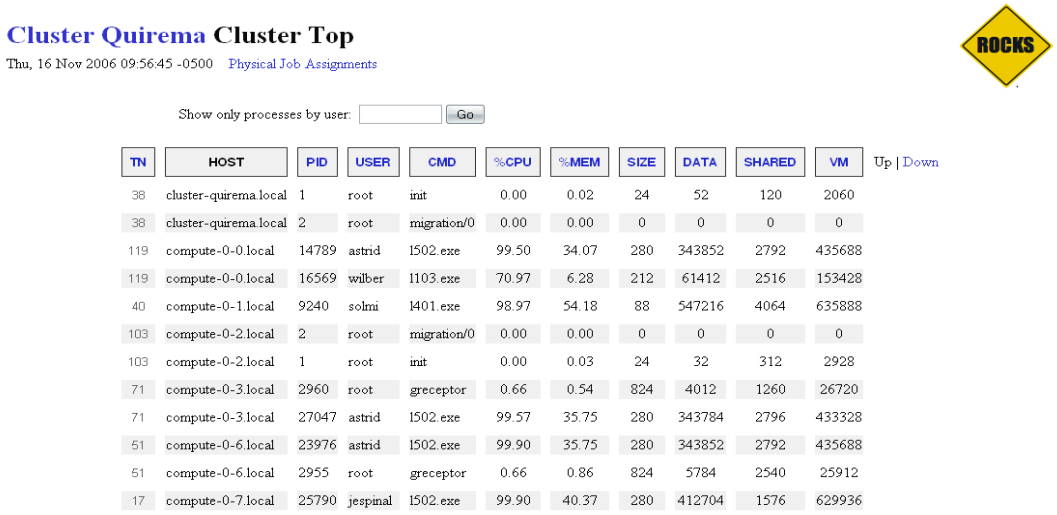
Thu, 16 Nov 2006 09:54:01 -0500 Physical Job Assignments

Show only processes by user: Go

TN	HOST	PID	USER	CMD	%CPU	%MEM	SIZE	DATA	SHARED	VM	Up | Down
78	compute-0-0.local	16569	wilber	l703.exe	99.90	6.64	272	64960	2588	158824	
38	compute-0-5.local	7527	eflorez	l502.exe	99.90	6.59	280	64852	2776	155112	
25	compute-0-6.local	23976	astrid	l502.exe	99.90	35.75	280	343852	2792	435688	
54	compute-0-8.local	17166	solmi	l1110.exe	99.90	61.68	156	630372	2652	723536	
17	compute-0-3.local	27047	astrid	l502.exe	99.57	35.75	280	343784	2796	433328	
78	compute-0-0.local	14789	astrid	l502.exe	99.56	34.07	280	343852	2792	435688	
77	compute-0-2.local	8257	wilber	l703.exe	99.50	6.77	272	66292	2572	158824	
15	compute-0-1.local	9240	solmi	l703.exe	99.03	53.27	272	539676	2424	632652	
36	compute-0-7.local	5308	jespinal	l1002.exe	98.91	41.41	260	423744	1280	875772	
36	compute-0-7.local	25790	jespinal	l502.exe	98.91	40.37	280	412704	1576	629936	
54	compute-0-8.local	2747	eflorez	l502.exe	97.52	6.61	280	65060	2776	155064	
33	compute-0-3.local	10134	sge	sge_execd	1.33	0.17	1148	312	1292	3920	

Figura 8. Lista de procesos en ejecución en el cluster ordenada ascendentemente por %cpu

Normalmente la lista de los procesos esta organizada en orden decreciente de %CPU. El orden puede modificarse usando los botones Up|Down que se encuentran en la parte superior derecha de la página. Es también posible cambiar el parámetro respecto al que se organiza la información en la lista de procesos usando los enlaces en los encabezados de cada columna. De particular interés puede ser organizar los procesos de acuerdo al nodo (en orden descendente) que permite conocer la manera como ese nodo (o el frontend mismo) esta siendo utilizado.



Cluster Quirema Cluster Top

Thu, 16 Nov 2006 09:56:45 -0500 Physical Job Assignments

Show only processes by user: Go

TN	HOST	PID	USER	CMD	%CPU	%MEM	SIZE	DATA	SHARED	VM	Up | Down
38	cluster-quirema.local	1	root	init	0.00	0.02	24	52	120	2060	
38	cluster-quirema.local	2	root	migration/0	0.00	0.00	0	0	0	0	
119	compute-0-0.local	14789	astrid	l502.exe	99.50	34.07	280	343852	2792	435688	
119	compute-0-0.local	16569	wilber	l103.exe	70.97	6.28	212	61412	2516	153428	
40	compute-0-1.local	9240	solmi	l401.exe	98.97	54.18	88	547216	4064	635888	
103	compute-0-2.local	2	root	migration/0	0.00	0.00	0	0	0	0	
103	compute-0-2.local	1	root	init	0.00	0.03	24	32	312	2928	
71	compute-0-3.local	2960	root	greceptor	0.66	0.54	824	4012	1260	26720	
71	compute-0-3.local	27047	astrid	l502.exe	99.57	35.75	280	343784	2796	433328	
51	compute-0-6.local	23976	astrid	l502.exe	99.90	35.75	280	343852	2792	435688	
51	compute-0-6.local	2955	root	greceptor	0.66	0.86	824	5784	2540	25912	
17	compute-0-7.local	25790	jespinal	l502.exe	99.90	40.37	280	412704	1576	629936	

Figura 9. Lista de procesos en ejecución en el cluster ordenada de acuerdo al nodo

- Cola de trabajos del SGE. La cola de trabajos en ejecución en el Scheduler se encuentra entre las facilidades más útiles que ofrece Ganglia. Esta se nos presenta como la alternativa gráfica del comando 'qstat' pero con una ventaja adicional.

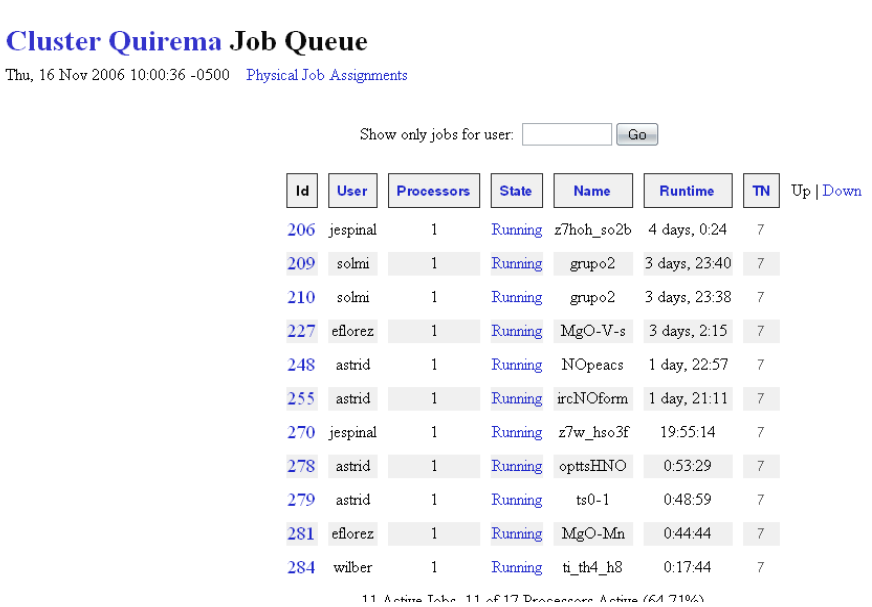


Figura 10. Lista de procesos en ejecución con SGE

La ventaja la ofrece la posibilidad de monitorear el estado de cada proceso, incluyendo información sobre el tiempo total de ejecución y un historial de uso del procesador sobre el que se ejecuta.

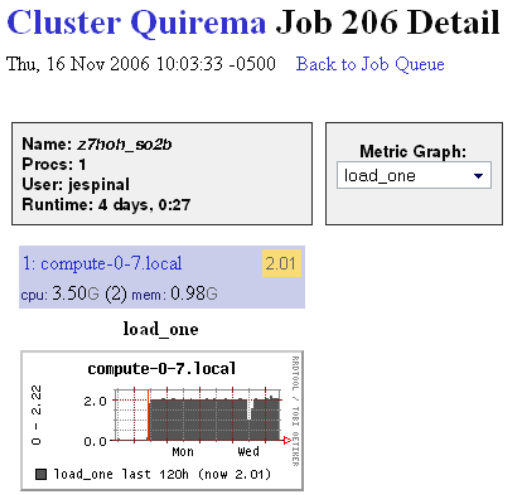


Figura 11. Información individual de un proceso en ejecución con SGE

Adicionalmente, usando la herramienta para el monitoreo de todos los procesos en el cluster (los que se ejecutan y los que no con SGE) se puede conocer gráficamente la manera como se asignan a cada nodo los distintos trabajos de SGE. Para ver esto diríjase al Link Cluster Top en la pagina principal de Ganglia y a continuación escoja el enlace Physical Job Assignments para ver un mapa de la asignación de los procesos.

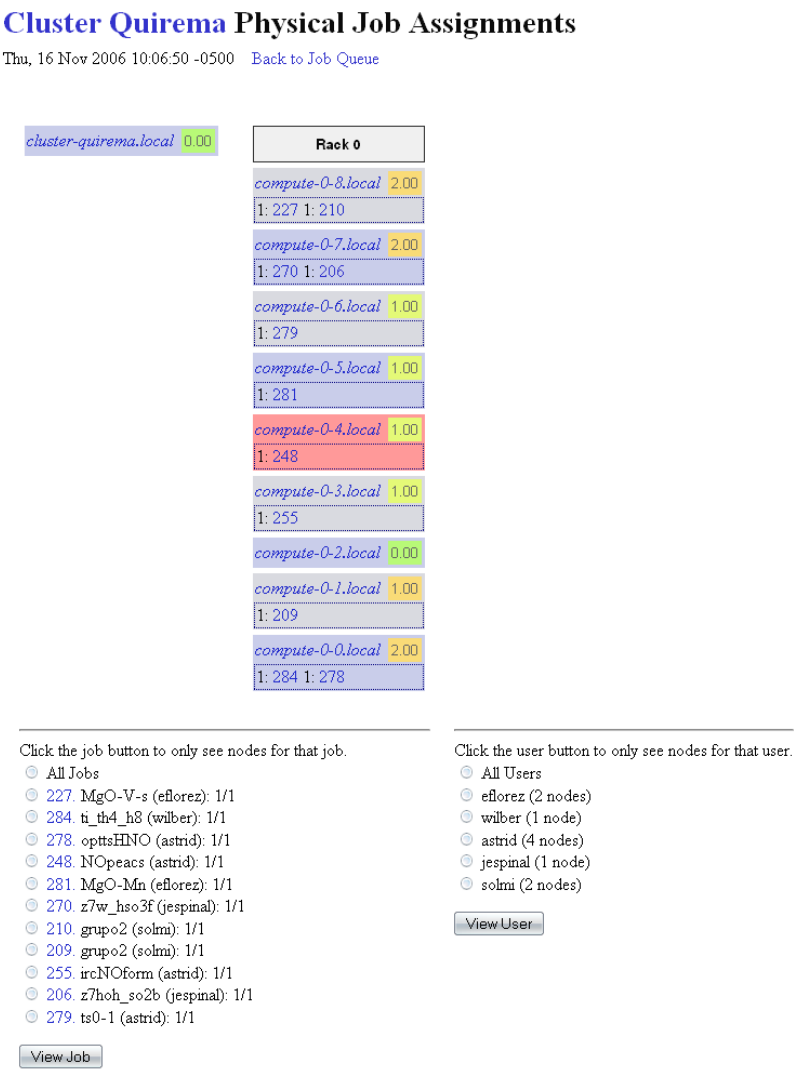


Figura 12. Physical Jobs Assignment

A través de esta página adicionalmente se puede conocer también información sobre cada procesos (o sobre todos) y sobre los usuarios que están ejecutándolos.

A.Síntesis de comandos

Comando	Explicación
# rpm -q gnuplot	Consulta la base de datos de rpms para saber si gnuplot esta instalado
# cluster-fork rpm -q gnuplot	Idem pero en todos los nodos
# rpm -Uvh gnuplot-3.7.3-2.i386.rpm	Instala gnuplot-3.7.3
# cluster-fork rpm -Uvh /share/apps/src/gnuplot-3.7.3-2.i386.rpm	Idem pero en todos los nodos
# cluster-fork which gnuplot	Determina la ubicación del binario gnuplot en los nodos
# ls -l /export/home/install/contrib/4.2.1/i386	Directorio donde se localizan los rpms que se desea agregar a la distribución
# cp -rf gnuplot-3.7.3-2.i386.rpm /export/home/install/contrib/4.2.1/i386/RPMS	Copia rpm en el directorio de donde la distribución lo tomara
# cp skeleton.xml extend-compute.xml	Crea el archivo de configuración extend-compute a partir de la plantilla skeleton
# rocks-dist dist	Reconstruye la distribución
# ls -ld rocks-dist/lan/i386/RedHat/RPMS/gnuplot-3.7.3-2.i386.rpm	Verifica la creación del enlace simbólico al paquete dentro de la distribución
# ssh c0-6 rm -rf /.rocks-release	Borra el archivo de bloqueo /.rocks.release
# ssh c0-6 /boot/kickstart/cluster-kickstart	Lanza el script de re instalación de Rocks en un nodo
# ./configure –prefix=/share/apps --COMPILED_BY="Jorge Zuluaga <zuluagajorge@gmail.com>"	Configura el instalador de Povray
# ./configure –help	Muestra todas las opciones recibidas por configure
# make install	Instala el paquete
# make -C /var/411 clean	Refresca los archivos de configuración del servicio 411
$ cp /share/apps/src/povray-3.6.1/scenes/advanced/sunsethf.pov .	Toma una de las escenas de pruebas proveídas con el paquete
$ povray +D +W640 +H480 +I sunsethf.pov	Dibuja “sunseth.pov” mostrando el resultado durante la ejecución y obligando a que la salida tenga resolución 640x480
$ ssh c0-5 /share/apps/bin/povray +D +W640 +H480 +I sunsethf.pov	Ejecuta remotamente povray
$ display sunsethf.png	Muestra el resultado de una ejecución
# ./configure --prefix=/share/apps	Configura el instalador de cualquier programa común a todos los nodos
$ gcc -I/share/apps/include -c testgsl.c -o testgsl.o	Compila (generar código objeto) de un programa que use GSL
$ gcc testgsl.o -L/share/apps/lib -lgslcblas -lgsl -o testgsl.out	Enlaza el código objeto con las bibliotecas
$ export LD_LIBRARY_PATH=/share/apps/lib	Fija la variable LD_LIBRARY_PATH
# ldconfig	Refresca la configuración del enlazador dinámico

Comando	Explicación
# cluster-fork ldconfig	Idem pero en todos los nodos
# service iptables restart	Reinicia el servicio iptables (seguridad, firewall)

Referencias

Linux y Linux Clustering:

1. Aprenda Linux como si estuviera en primaria, http://mat21.etsii.upm.es/ayudainf/aprendainf/Linux/Linux.pdf.
2. William Gropp y otros, Beowulf Cluster Computing with Linux, Second Edition (Scientific and Engineering Computation)

Rocks and rolls:

3. Rocks User Guide. http://www.rocksclusters.org/rocks-documentation/4.2.1
4. Rolls User Guide:
   Ganglia: http://www.rocksclusters.org/roll-documentation/ganglia/4.2.1,
   SGE: http://www.rocksclusters.org/roll-documentation/sge/4.2.1
5. Tutoriales y otras guías en Rocks:
   http://www.rocksclusters.org/rocksapalooza/2006

Servicios:

6. NFS Howto. http://nfs.sourceforge.net/nfs-howto
7. Autofs howto. http://www.linux-consulting.com/Amd_AutoFS/autofs.html

Herramientas:

8. Manual de Gnu Scientific Library.
   http://www.gnu.org/software/gsl/manual/html_node/index.html
9. Sitio oficial de PovRay.
   http://www.poray.org

Otras lecturas:

10. Hentosh, Robert. Whitepaper “Linux Enumeration of Nicks”
    http://linux.dell.com/files/whitepapers/nic-enum-whitepaper-v2.pdf