CONSTRUCCIÓN DE UN CLUSTER PARA CÁLCULO PARALELO MPI

Jose Manuel Martínez García

Ramón Ángel Fernández Díaz

Escuela de Ingenierías Industrial e Informática

Universidad de León

Edición 2011

INTRODUCCIÓN

El presente documento explica la instalación y configuración de un cluster de cálculo paralelo *MPI totalmente basado en software libre.

Aunque los procedimientos aquí indicados se basan en arquitecturas hardware específicas encontradas en el laboratorio del IAF de la EIII y versiones de software concretas, podría aplicarse a entornos más amplios.

LICENCIA Y PROPIEDAD

INDICE

OBJETIVOS Y PLAN DE TRABAJO

Este proyecto trata de lograr dos objetivos principales:

1. Construir un cluster de cálculo paralelo *MPI totalmente basado en software libre y arquitecturas estándar.
   - Definición de configuración de hardware óptima a partir de recursos limitados.
   - Definición de la configuración de red óptima a partir de recursos limitados.
   - Configuración del software de sistema y de cálculo.
   - Caracterización y optimización del cluster obtenido desde un punto de vista didáctico.

2. Generar la documentación necesaria con la experiencia para poder disponer de material didáctico en diferentes asignaturas del Grado en Informática o titulaciones de temática similar.

Plan de trabajo

Se compone de cuatro partes:

1.- Introducción a la teoría de supercomputación basada en cálculo paralelo respaldado el clusters.

2.- Instalación de hardware, software y herramientas de monitorización.

3.- Instalación de liberías y utilidades de cálculo parelelo y ejecución de programas de test.

4.- Pruebas de caracterización de la plataforma, y prácticas de mejora de rendimiento.

TEORÍA DE *CLUSTERS* Y CALCULO PARALELO

¿Qué es la Supercomputación?

Desde los primeros días de las computadoras, allá por los años 50, se ha observado su potencial como herramienta para hacer avanzar la ciencia.

Casi desde entonces las Universidades y centros de investigación de todo el mundo han sido importantes compradores de computadoras. El uso de las mismas ha correspondido sobre todo a científicos que las han utilizado para realizar de forma rápida cálculos complejos y repetitivos que de otra forma les llevarían años.

Pero la investigación y la ciencia nunca se ha conformado con la *oferta* que la tecnología del momento podía ofrecerles en cuanto a potencia de cálculo, capacidad de almacenamiento de información o velocidad de transmisión de datos. Su *demanda* siempre ha sido superior.

La constante lucha por exprimir la tecnología para obtener de ella lo que ni tan siquiera sus creadores pensaron, para dar respuesta a esa demanda, ha terminado convirtiéndose en una *ciencia* en sí misa. Es lo que se conoce como *Supercomputación*.

La supercomputación ya está reconocida, de facto, como una disciplina completa de la informática y como la herramienta más poderosa de la ciencia y la investigación hoy. Resulta difícil encontrar una disciplina en la que no esté presente de alguna forma la supercomputación; por ejemplo:

- Simulación de procesos químicos.
- Simulación de choques.
- Simulación de procesos industriales.
- Cálculos para astronomía y astrofísica.
- Secuenciación de material genético.
- Procesado de grandes cantidades de información.

Todas estas y muchas más, tienen en la supercomputación una de sus herramientas principales.

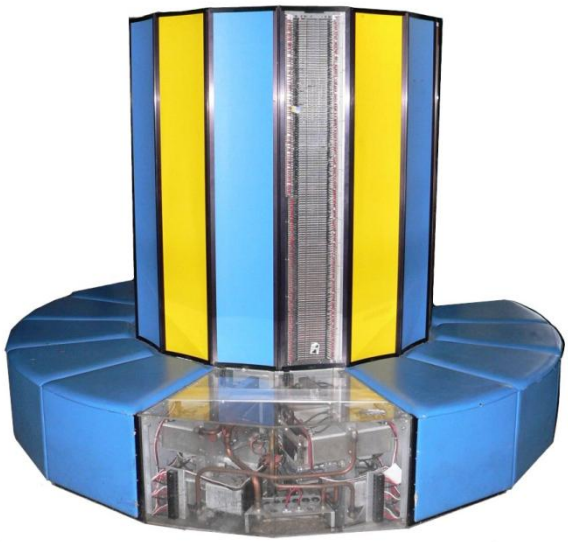
Evolución de la Supercomputación

La Supercomputación ha pasado por multitud de fases en las últimas décadas. Desde ser una disciplina tabú y solo para *elegidos* hasta convertirse casi en una *comodity* en la actualidad.

A continuación se hará un breve repaso de la Supercomputación desde el punto de vista de los supercomputadores. El objetivo es llegar a entender el porqué de como se construyen hoy en día.

En principio las supercomputadoras se construían solo bajo demanda, desde cero y como un producto aislado y muy enfocado a las especificaciones del comprador. Solo las grandes compañías de la industria como IBM eran capaces de construir estas máquinas, cuyo precio superaba con frecuencia las 9 cifras. El diseño de la máquina se ajustaba a unos objetivos de cálculo muy claros y podía durar años. El resultado era una máquina muy especializada.

En 1972 Seymour Cray, viendo el negocio que podría suponer satisfacer la demanda de supercomputadores, funda una compañía (Cray Research) con el objetivo de construir supercomputadores en serie y de propósito ligeramente más general. Sus planes se materializan en 1979 con la venta del primer Cray-1, la primera supercomputadora comercial. Su diseño estético se ha convertido en el icono de la supercomputadoras.



Cray-1

Cray se convirtió en la pionera de la supercomputación comercial, vendiendo docenas de máquinas Cray-1 y sus descendientes a lo largo de 30 años hasta hoy. Sin embargo, y aunque mantiene algunas de sus creaciones en la zona alta del Top500, ha pasado a ser una compañía de *segunda fila*. Las razones de esta caída no están en la propia compañía, sino en el cambio en el mundo de la supercomputación.

A finales de los años 80 y principios de los 90 comprar una supercomputadora a IBM, CRAY o DEC era sencillo salvo por una cosa: Era enormemente caro. Además, tenían otras limitaciones intrínsecas a su arquitectura:

- Eran máquinas tan especializadas que solamente podían ser usadas en una tarea específica.
- Seguían muy supeditadas a la velocidad de la tecnología del momento.
- No eran escalables. Cuando eran demasiado lentas no podían ser ampliadas y era necesario sustituirlas.
- Funcionaban con protocolos propietarios y cerrados, lo que hacía que el comprador quedara *atado* su fabricante.
- Eran difíciles de programar y al no usar protocolos abiertos los programas eran interoperables entre ellas.

Para intentar solventar estos problemas, especialmente el económico, surgiría la idea de utilizar componentes menos especializados, casi de consumo para crear supercomputadoras sumando pequeñas computadoras menos potentes. Así surgieron los clusters de cálculo, que es la forma en que se construyen las supercomputadoras hoy en día.

http://en.wikipedia.org/wiki/Cray-1

http://en.wikipedia.org/wiki/Cray_Research

Definición de cluster

Aunque no existe una definición "oficial", se considera un cluster (en el ámbito de las computadoras) como un conjunto de ordenadores interconectados entre sí que trabajan de forma colaborativa para solucionar un problema como si de uno solo se tratase. Cada uno de los ordenadores que forman el cluster se conoce como "nodo". La gran diferencia con otros grandes sistemas es que al haber múltiples nodos, cada uno ejecuta su propia instancia de sistema operativo, que debe ser similar en todos ellos aunque no es así en todos los casos.

Un cluster es un sistema distribuido, normalmente débilmente acoplado, donde se delega gran parte de las tareas de paralelización y distribución de carga a las capas de software superiores.

http://es.wikipedia.org/wiki/Cluster_%28inform%C3%A1tica%29

Tipos de clusters

Existen clusters de distintos tipos; y atendiendo a su definición se suelen encontrar tantos como tipos de problemas se trata de resolver. Por ejemplo:

- **Clusters de cálculo**: Intentan conseguir una máquina de cálculo más potente sumando las capacidades discretas de los nodos que lo componen (caso tratado).
- **Clusters de alta disponibilidad**: Tratan de mantener operativo un servicio minimizando los daños producidos por errores en el software o en hardware. Normalmente se construyen usando el doble de recursos de los necesarios para hacer operativo un servicio, de forma que si un recurso se avería, existe un en modo *spare* que no sustituye inmediatamente.
- **Clusters de balanceo de carga**: Cuando una única máquina no es capaz de atender un número de peticiones de servicio, se agregan más máquinas similares de forma que las peticiones se distribuyan entre todos los nodos que conforman el cluster.

El primer cluster para cálculo

En el año 1994, Donald Becker y Thomas Sterling construyeron la primera máquina de cálculo masivo con estructura de cluster. Dispusieron 16 ordenadores personales en red y reescribieron algunos de sus programas monohilo para que eje ejecutaran en múltiples hilos y así poder distribuir el cálculo entre los 16 nodos.

A este primer cluster (oficial) de cálculo paralelo se le conoce como "*Cluster Beowulf*" y es a los supercomputadores actuales lo que en el año 1981 el IBM PC a los ordenadores personales de hoy en días (o el primer MAC en 1984). Fue el primero, el pionero y definió el concepto de los que habrá sido la estructura fundamental de la supercomputación durante los siguientes veinte años.

La construcción de los primeros Beowulf respondían, en gran medida, a la necesidad de sus creadores de acceder a una potencia cálculo a la que económicamente les resultaba imposible acceder.

En aquella época, las grandes máquinas de los principales fabricante (IBM, Cray…) dominaban el mercado. Ofrecían potencias suficientes pero a un precio enorme. Además, sus diseños y protocolos eran propietarios y cerrados.

La idea de usar computadoras personales para crear una máquina más potente con la suma de la potencia individual de cada una no se había planteado en serio hasta entonces. Incluso después tuvieron que pasar años hasta que la industria entendió que aquel era el futuro de la supercomputación.

http://en.wikipedia.org/wiki/Beowulf_cluster

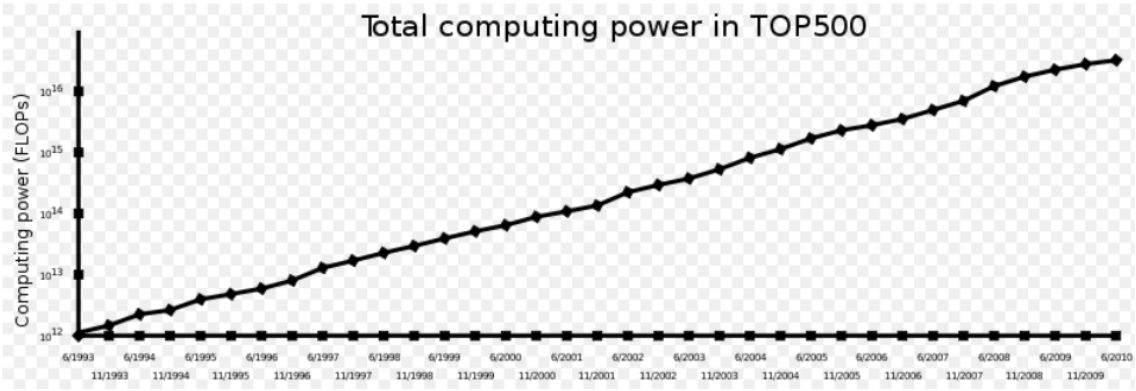
http://www.beowulf.org

Top 500

En Junio del año 2003, un grupo de Universidades e institutos de investigación alrededor del mundo, acuerdan crear “top500.org”.

Es una iniciativa para unir los datos de potencia de los supercomputadores más potentes del mundo. Cualquier usuario o institución puede enviar los datos de potencia de sus máquinas para compararlas con las existentes.

El ranking top500 no es más que un número de TeraFlops optenido con test Linpack y dista enormemente de ser un dato objetivo sobre la verdadera utilidad práctica de estas supermáquinas. Sin embargo es interesante para observar su evolución.



Evolución de la potencia total en el Top 1

De gráficas como la anterior se desprende que la evolución en potencia instalada que computa para el Top500 tiene un crecimiento lineal.

Se puede consultar la lista, con mucha otra información interesante sobre supercomputación, en la URL www.top500.org

La lista se actualiza cada seis meses con los resultados de los test de rendimiento enviados por los dueños de las supercomputadoras. En la actualización de Noviembre de 2011, la primera del ranking era una máquina Japonesa llamada 'K', que ostenta el honor de ser la primera máquina en pasar de una potencia de cálculo pico de 10 PetaFlops.

http://top500.org

http://*en.wikipedia.org/wiki/**K**_computer*

http://www.aics.riken.jp/en/kcomputer/

Futuro de la supercomputación

En el momento actual se está viviendo una auténtica revolución en la supercomputación. Normalmente este es un campo en constante cambio y la frase anterior puede ser dicha, sin temor a equivocarse, en cualquier momento. Pero ciertamente desde el año 2008 se percibe un cambio en el horizonte. A finales de 2011 ese cambio está tan cerca que ya está ocurriendo.

Durante los primero años del Siglo XXI, parecía clara cual debía ser la arquitectura de los supercomputadores y su programación. En aquel momento los supercomputadores se construían como clusters. Los nodos de los clusters contenían componentes electrónicos de arquitecturas más o menos generalistas (Intel X86 o Itanium, PowerPC, Sparc, etc), procesadores programables de propósito general y programas escritos en MPI que paralelizaran los cálculos.

Mientras todo esto "evolucionaba-vegetaba", en el mercado de consumo se producía una pequeña revolución: El hardware para entretenimiento. Y algunas empresas de tamaño medio, como NVidia o ATI, comenzaron a construir una subarquitectura específica (procesadores, buses y memorias) para ejecutar programas de juegos.

Con la evolución de estos nuevos sistemas la industria del entretenimiento comenzó a explotar, demandando actualizaciones sobre ellos para mejorar, principalmente, el aspecto gráfico de los juegos. Estaban, casi sin quererlo, preparando el cambio más importante en la supercomputación en veinte años.

El hardware de las tarjetas gráficas, videoconsolas, etc. Es enormemente simple. La idea es similar a la introducción de las arquitecturas RISC años atrás; solo hacen unas pocas operaciones, pero tan especializadas que pueden hacerlas a una velocidad hasta dos órdenes de magnitud mayor que un procesador programable de propósito general. Además, al ser mucho más pequeños (por el menor número de componentes que necesitan), pueden incorporarse muchas unidades de proceso en el mismo procesador (GPU).

Las operaciones que ejecuta una GPU están directamente relacionadas con el manejo de imágenes y gráficos, que a nivel de programación supone el manejo de grandes matrices de datos (rotar, transponer, multiplicar, invertir…). Pero la mayoría de programas de supercomputación o HPC basan su programación precisamente en el manejo de grandes matrices de números, de forma que se comenzó a pensar y a investigar la posibilidad de

programar las aplicaciones HPC con las mismas librerías de desarrollo que los videojuegos ya que hacen las mismas operaciones. El resultado ha sido espectacular.

Actualmente, cuatro de los cinco primeros supercomputadores del 'Top500' (menos el primerio, curiosamente) están construidos principalmente con GPUs y uno de ellos ya estuvo el primero del 'Top500' en la primera mitad de 2011. Fabricantes de GPU's, principalmente NVIDIA y ATI, han desarrollado librerías de desarrollo HPC basadas en sus tarjetas gráficas.

En el mundo de la supercomputación ya nadie duda que una parte cada vez más importante de las grandes máquinas de cálculo que se construirán en los próximos cinco años estarán formadas por GPUs.

Software libre y supercomputación

El software de código (y no necesariamente el software libre) ha estado ligado a la supercomputación desde sus inicios. De hecho continua estándolo y parece probable que siga así durante bastante tiempo aún.

El motivo es que tienen ascendentes comunes: La ciencia y la investigación.

Los genios que crearon los primeros clusters y que necesitaban las primeras supercomputadoras eran investigadores y científicos. Las personas que han escrito la mayoría de programas de cálculo masivo han sido investigadores y científicos. Ya que en ese mundo la compartición de código es la práctica habitual, no es de extrañar que SW Libre y supercomputación caminen de la mano.

ESQUEMA FUNCIONAL Y DE SERVICIOS

Para llevar a cabo la construcción del cluster se han identificado los siguientes bloques funcionales

- Sistema de almacenamiento compartido.

- Base de datos de usuarios compartida.

- Diferentes nodos de cálculo aislados.

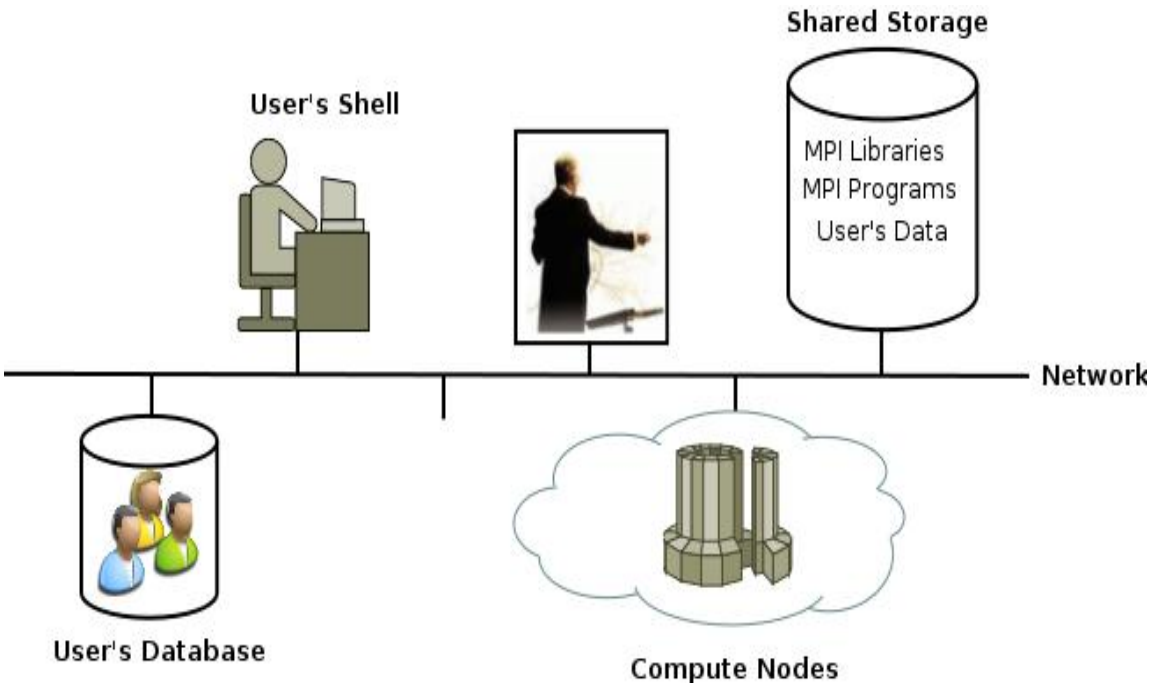
- Red de comunicaciones.

- Librerías de procesamiento paralelo.

- Software compilado para funcionar en paralelo.

- Sistema de orquestación para lanzamiento y control de los trabajos (gestor de colas).

- Shell para usuarios.



INVENTARIO DE HARDWARE

Este proyecto puede ser realizado con multitud de hardware diferente. Obviamente, a mayor potencia de los componentes discretos, mayor será el rendimiento global del sistema, pero para ese caso se ha utilizado:

4 Ordenadores de sobre mesa con procesador Intel Pentium III (666Mhz), 256 MB de memoria SD-Ram y un disco duro IDE de 40GB.

Todos están interconectados a través de una interfaz Ethernet 100mbs y un switch HP Procurve 2420.

Las 4 máquinas propuestas (manager, storage, nodo1 y nodo2) comparten idéntica arquitectura Intel x86 de 32 bits. La única diferencia está en la máquina "storage", que dispone de un disco duro adicional para los sistemas de ficheros NFS.

A continuación se muestra la salida de los comandos "df, free" y del contenido de "/proc/cpuinfo" de algunos de ellos.

[root@caligula test_mpi]# df

```
S.ficheros        Bloques de 1K   Usado    Dispon Uso% Montado en
/dev/hda2           18946404   7624508  10343940  43% /
/dev/hda1             101086     11306     84561  12% /boot
tmpfs                 127704         0    127704   0% /dev/shm
/dev/hdc1           19619800    304016  18319152   2% /opt/sge
storage:/software   38456320   1902144  34600672   6% /software
storage:/home       38464384    180256  36330208   1% /home
```

[root@caligula test_mpi]# free -m

```
 total       used       free     shared    buffers     cached
Mem:           249        241          7          0          3        102
-/+ buffers/cache:        136        113
Swap:          258          0        258
```

[root@caligula test_mpi]# cat /proc/cpuinfo

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 8
model name      : Pentium III (Coppermine)
stepping        : 3
cpu MHz         : 668.965
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
```

flags : fpu vme de pse tsc msr pae mce cx8 mtrr pge mca cmov pat pse36 mmx fxsr sse up
bogomips : 1337.93

root@caligula test_mpi]# ssh storage df
S.ficheros Bloques de 1K Usado Dispon Uso% Montado en
/dev/hda3 18697532 2799436 14932992 16% /
/dev/hda1 101086 11307 84560 12% /boot
tmpfs 127708 0 127708 0% /dev/shm
/dev/hdb1 38456308 1902160 34600648 6% /software
/dev/hdb2 38464372 180272 36330196 1% /home

[root@caligula test_mpi]# ssh storage free -m
total used free shared buffers cached
Mem: 249 241 8 0 20 156
-/+ buffers/cache: 64 185
Swap: 509 0 509

[root@caligula test_mpi]# ssh storage cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 8
model name : Pentium III (Coppermine)
stepping : 3
cpu MHz : 700.068
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 mtrr pge mca cmov pat pse36 mmx fxsr sse up
bogomips : 1400.13

NODOS

[root@caligula test_mpi]# ssh nodo1 df
S.ficheros Bloques de 1K Usado Dispon Uso% Montado en
/dev/mapper/VolGroup00-LogVol00
18664164 5170872 12529916 30% /
/dev/hda1 101086 12013 83854 13% /boot
tmpfs 127708 0 127708 0% /dev/shm
storage:/software 38456320 1902144 34600672 6% /software
storage:/home 38464384 180256 36330208 1% /home

caligula:/opt/sge 19619808 304000 18319168 2% /opt/sge

[root@caligula test_mpi]# ssh nodo1 free -m

	total	used	free	shared	buffers	cached
Mem:	249	226	22	0	32	147
-/+ buffers/cache:		47	202			
Swap:	511	0	511			

[root@caligula test_mpi]#

[root@caligula test_mpi]# ssh nodo1 cat /proc/cpuinfo

processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 8
model name : Pentium III (Coppermine)
stepping : 3
cpu MHz : 665.022
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 mtrr pge mca cmov pat pse36 mmx fxsr sse up
bogomips : 1330.04

INVENTARIO DE SOFTWARE

Se utilizarán los siguientes paquetes de software, intentando maximizar el uso de open-source.

- Sistema operativo base del cluster (nodos de cálculo, nodos de almacenamiento y cabecera): GNU/Linux CentOS 5.5
- Sistema de almacenamiento compartido: NFS
- Base de datos de usuarios: OpenLDAP
- Red de comunicaciones: Ethernet 100mps / TCP-IP
- Librerías de procesamiento paralelo: OpenMPI
- Gestor de colas: “No contemplado en esta versión”
- Shell para usuarios: BASH

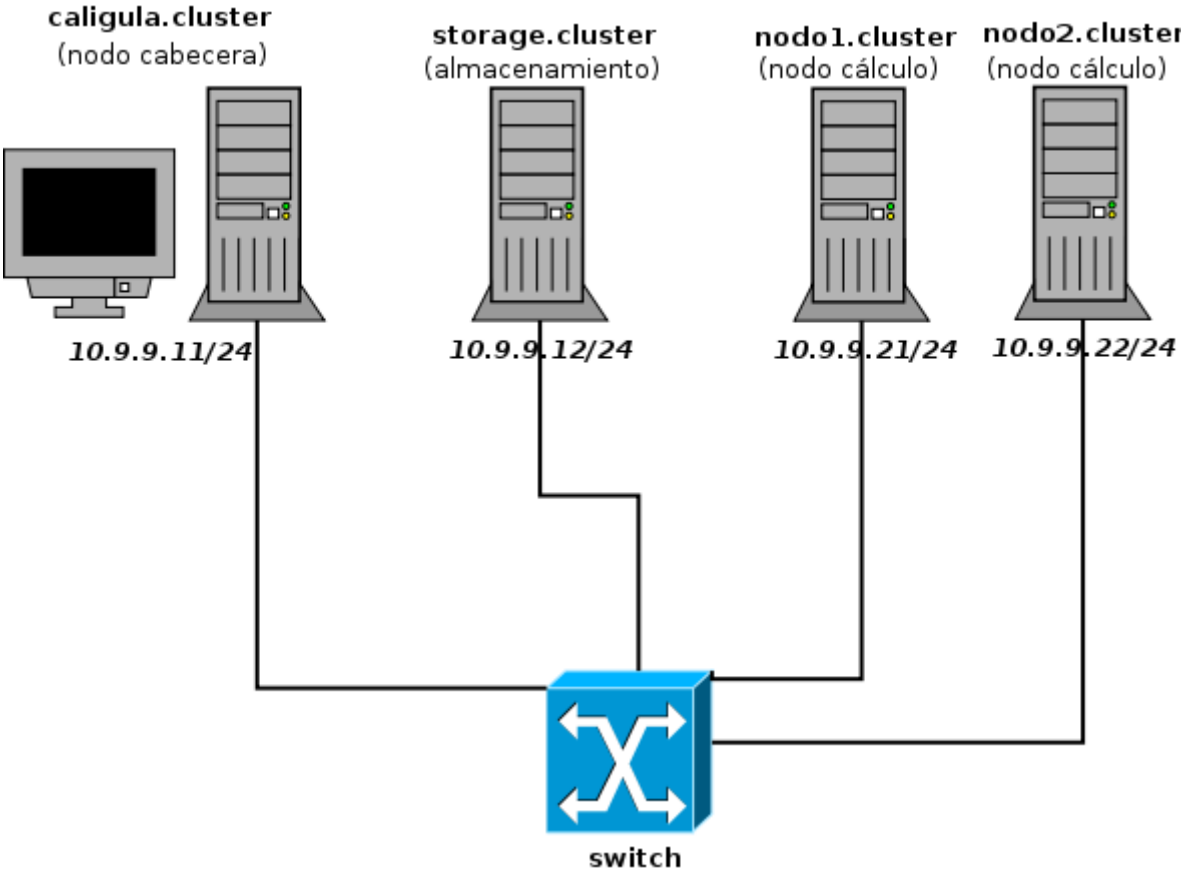
ESQUEMA DE RED Y DIRECCIONAMIENTO

El cluster contará con 4 nodos interconectados por ethernet a través de un switch:

- 1x nodo cabecera: Shell de usuarios, gestor de colas y lanzador de trabajos, management y monitorización.
  Hostname: caligula.cluster
  Dirección IP: 10.9.9.11
- 1x nodo de almacenamiento NFS.
  Hostname: storage.cluster
  Dirección IP: 10.9.9.12
- 2x nodos de cálculo MPI:
  Hostname: «nodo1.cluster», «nodo2.cluster»
  Dirección IP: 10.9.9.21, 10.9.9.22

Los datos de la red IP son los siguientes:

- Nombre: «cluster»
- IP: 10.9.9.0
- Netmask: 255.255.255.0 (/24)



INSTALACIÓN DEL HARDWARE

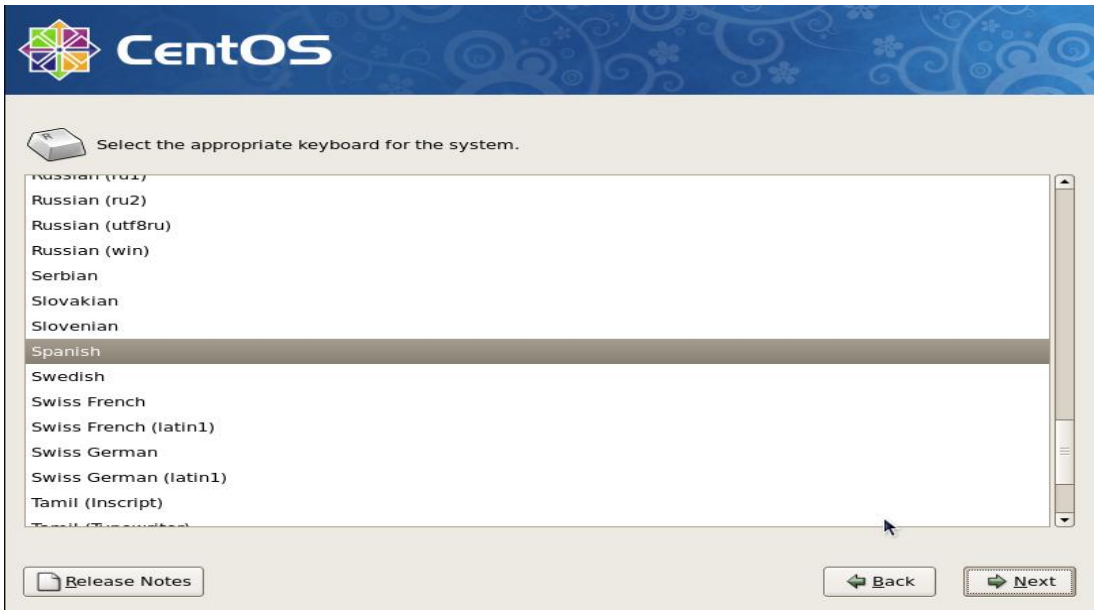
No se describe el proceso de instalación del HW.

El objetivo es crear una infraestructura de red basada en el esquema de red del punto anterior.

SOFTWARE BASE: SISTMA OPERATIVO DE TODOS LOS NODOS

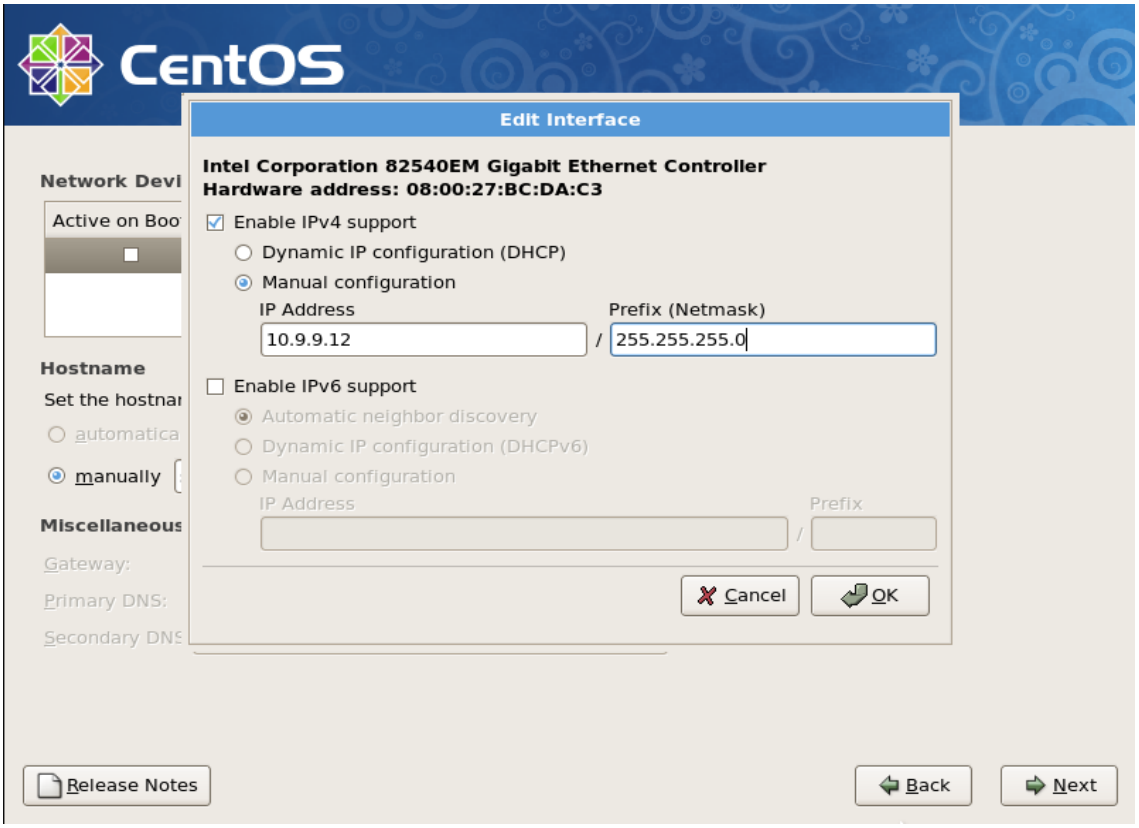
Todos los equipos del cluster parten de una instalación básica estándar de la distribución CentOS 5.5 compilada para arquitectura i586.

Se recomienda la instalación del sistema en idioma Inglés.

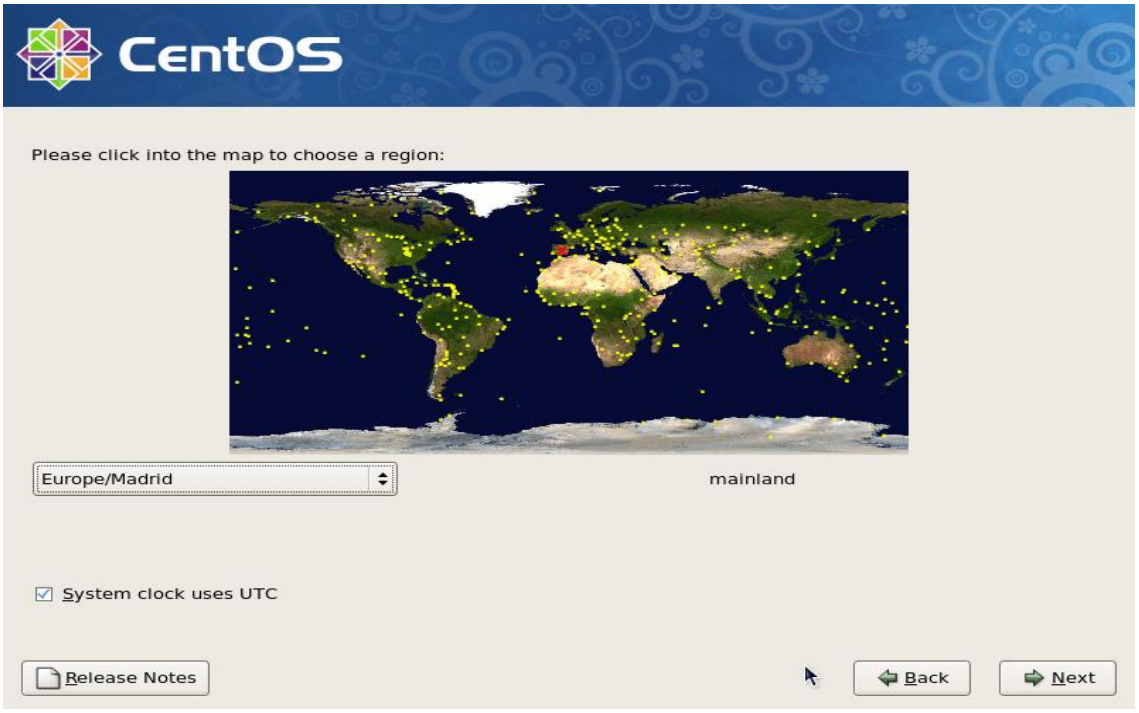

CentOS
Select the appropriate keyboard for the system.
Russian (ru2)
Russian (utf8ru)
Russian (win)
Serbian
Slovakian
Slovenian
Spanish
Swedish
Swiss French
Swiss French (latin1)
Swiss German
Swiss German (latin1)
Tamil (Inscript)
Release Notes
Back
Next

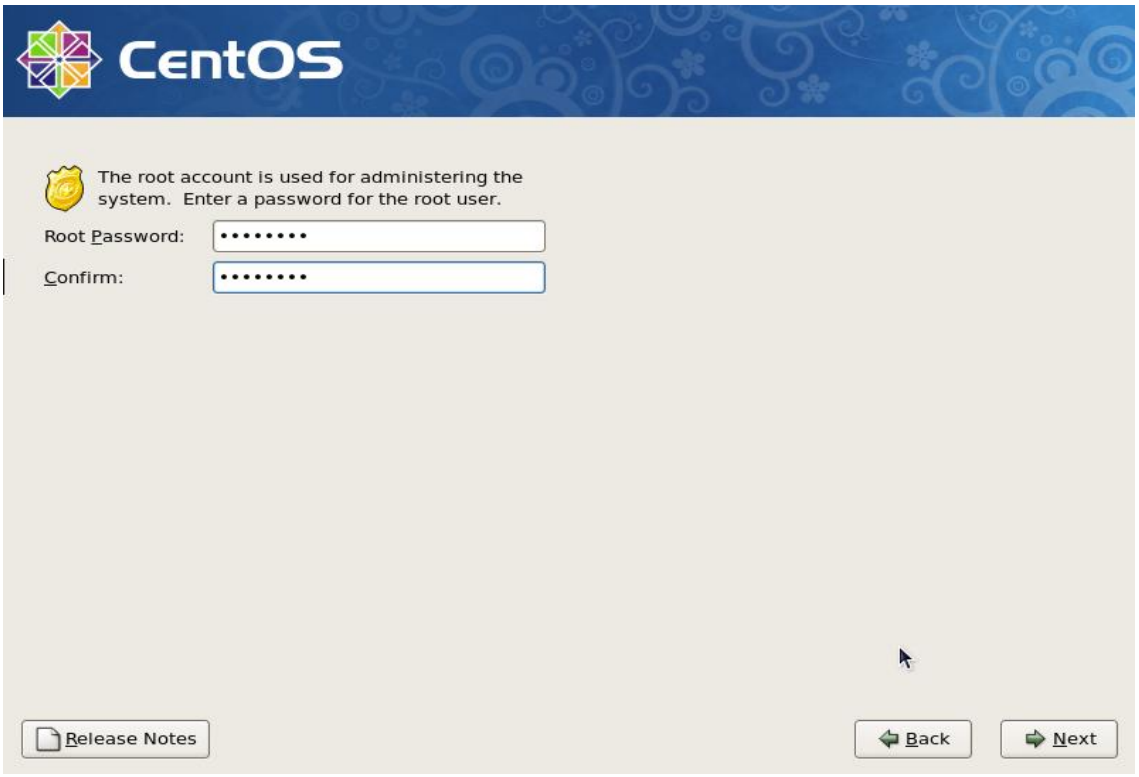

CentOS
Install CentOS
Choose this option to freshly install your system. Existing software and data may be overwritten depending on your configuration choices.
Upgrade an existing installation
Choose this option if you would like to upgrade your existing CentOS system. This option will preserve the existing data on your drives.
The following installed system will be upgraded:
CentOS 5.5 (/dev/VolGroup00/LogVol00)
Release Notes
Back
Next

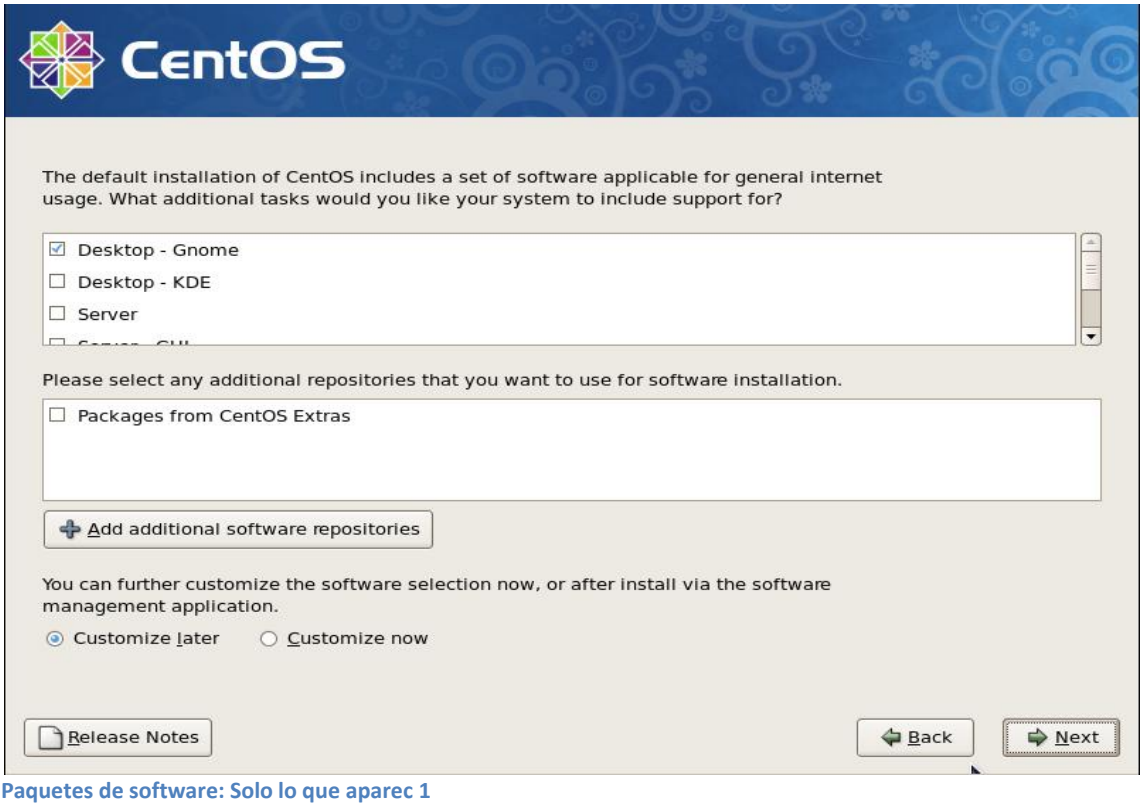

CentOS
Installation requires partitioning of your hard drive.
By default, a partitioning layout is chosen which is
reasonable for most users. You can either choose
to use this or create your own.
Remove linux partitions on selected drives and create default layout.
Encrypt system
Select the drive(s) to use for this installation.
sda 8189 MB ATA VBOX HARDDISK
Advanced storage configuration
Review and modify partitioning layout
Release Notes
Back
Next


CentOS
Edit Interface
Intel Corporation 82540EM Gigabit Ethernet Controller
Hardware address: 08:00:27:BC:DA:C3
Enable IPv4 support
Dynamic IP configuration (DHCP)
Manual configuration
IP Address
Prefix (Netmask)
10.9.9.12
255.255.255.0
Enable IPv6 support
Automatic neighbor discovery
Dynamic IP configuration (DHCPv6)
Manual configuration
IP Address
Prefix
Cancel
OK
Network Devi
Active on Boo
Hostname
Set the hostnar
automatica
manually
Miscellaneous
Gateway:
Primary DNS:
Secondary DNS
Release Notes
Back
Next


CentOS
Network Devices
Active on Boot
Device
IPv4/Netmask
IPv6/Prefix
Edit
eth0
10.9.9.12/24
Disabled
Hostname
Set the hostname:
automatically via DHCP
manually
storage.cluster
(e.g., host.domain.com)
Miscellaneous Settings
Gateway:
Primary DNS:
Secondary DNS:
Release Notes
Back
Next

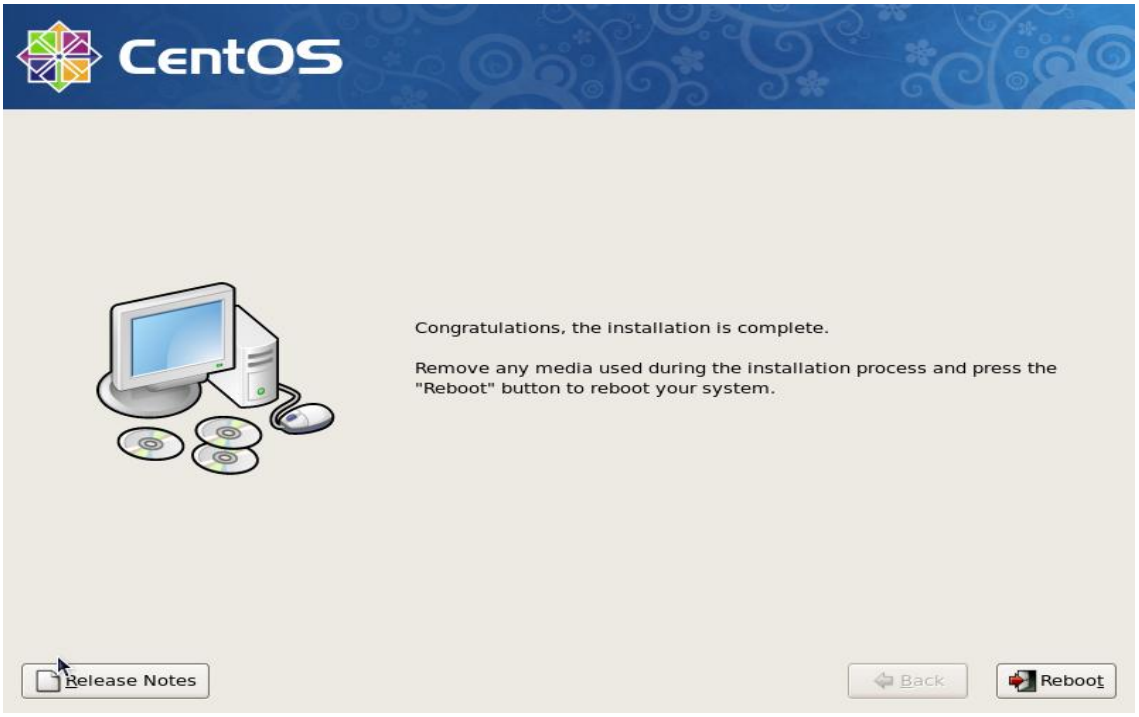

CentOS
Please click into the map to choose a region:
Europe/Madrid
mainland
System clock uses UTC
Release Notes
Back
Next





Paquetes de software: Solo lo que aparec 1

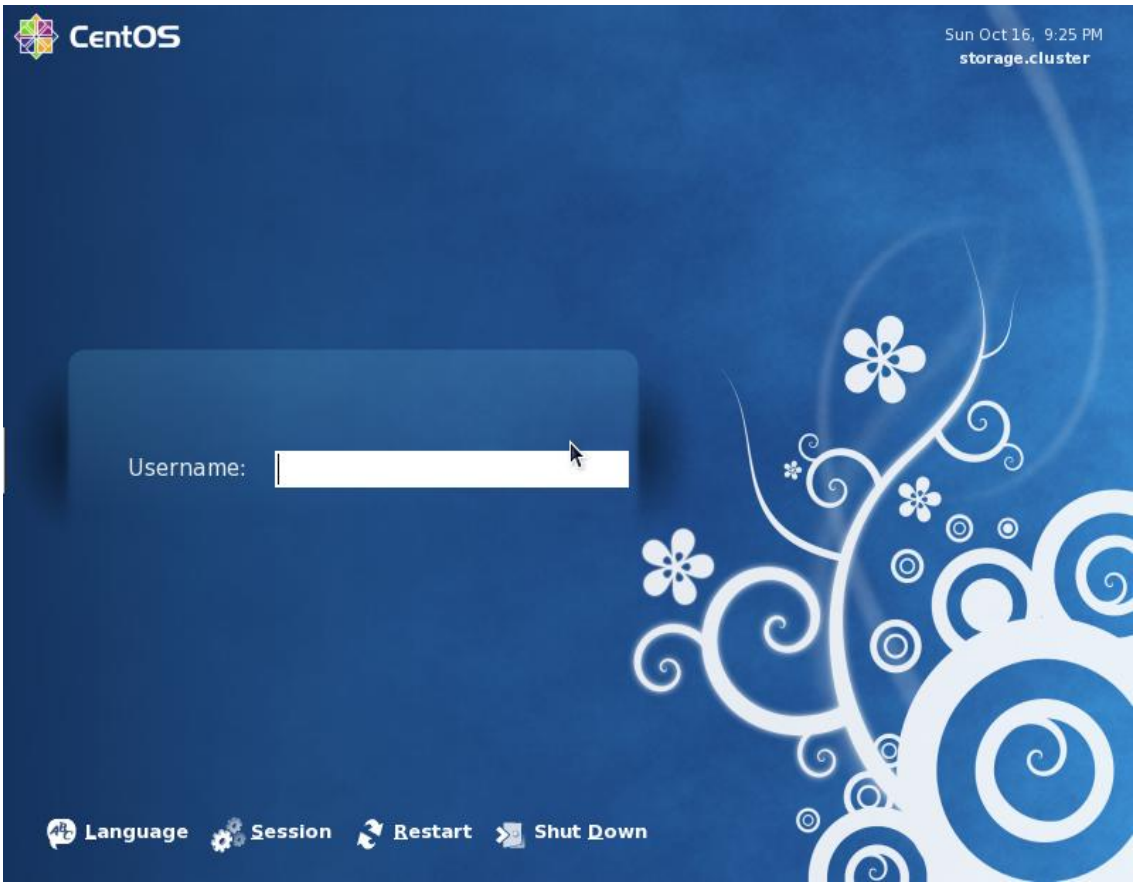

CentOS
Welcome to CentOS 5 !
Thank you for installing CentOS 5.
CentOS is an enterprise-class Linux Distribution derived from sources freely provided to the public by a prominent North American Enterprise Linux vendor.
CentOS conforms fully with the upstream vendors redistribution policy and aims to be 100% binary compatible. CentOS mainly changes packages to remove upstream vendor branding and artwork.
http://www.centos.org/
Installing glibc-common-2.5-49.i386 (63 MB)
Common binaries and locale data for glibc
Release Notes
Back
Next


CentOS
Congratulations, the installation is complete.
Remove any media used during the installation process and press the "Reboot" button to reboot your system.
Release Notes
Back
Reboot

...REBOOT...

DESACTIVAR FIREWALL 1

DESACTIVAR SELINUX 1



Configuraciones comunes "post-instalación"

Además, todos los nodos deben compartir los siguientes archivos de configuración, que deben ser idénticos en todos ellos:

/etc/hosts

127.0.0.1	*localhost*
10.9.9.11	*caligula*
10.9.9.12	*storage*
10.9.9.21	*nodo1*
10.9.9.22	*nodo2*

/etc/resolv.conf

search	*caligula*

Desactivar IPv6.

Solo es necesario crear el usuario «root»

Configurar correctamente el gestor de paquetes: Ya que será necesario instalar multitud de paquetes de software, se recomienda configurar la herramienta YUM para que los gestione e instale así como sus dependencias.

En caso de que el nodo tenga salida directa a Internet, configurar YUM para que use los repositorios oficiales de CentOS (u otros) directamente vía http:

Consultar: *http://www.centos.org/docs/5/html/yum/*

En caso de no tener salida a Internet, se puede agregar como repositorio el medio de instalación (ISO, CD, DVD…), que contiene por defecto casi todos los necesarios:

Consultar: *http://www.cyberciti.biz/tips/redhat-centos-fedora-linux-setup-repo.html*

INSTALACIÓN/CONFIGURACIÓN DEL NODO DE ALMACENAMIENTO: «STORAGE.CLUSTER»

Partiendo de que el equipo que realizará las funciones tiene instalado el sistema operativo estándar y la configuración de red (IP, Netmask, Hostname) adecuada, se realizan los siguientes pasos.

Planificación del almacenamiento

Aunque no es obligatorio, sí es buena idea que toda la información que se va a exportar vía NFS esté en un disco duro o partición separada del resto del sistema. En este caso, se ha utilizado un disco completo.

Instalación del servicio y configuración

Se exportarán dos directorios:

/storage/home <- Directorios $HOME de los usuarios del cluster

/storage/software <- Directorio donde se almacenará todo el software paralelo para que esté disponible a todos los usuarios en todos los nodos

Crear la estructura:

```
mkdir  -p /storage/home
mkdir -p /storage/software
```

Instalar el demonio NFS

```
yum install  nfs-utils portmap
```

Exportar los directorios elegidos vía NFS. Se configura a través del archivo «/etc/exports»

/etc/exports

```
/storage/software       10.9.9.0/24(rw,no_root_squash)
/storage/home           10.9.9.0/24(rw,no_root_squash)
```

La sintaxis de este archivo es, básicamente:

DIRECTORIO_A_EXPORTAR[ESPACIO]RED_PERMITIDA(OPCIONES)

Ejecutar: *exportfs -a*

Arrancar los demonios y configurar su arranque automático

service nfs restart

service portmap restart

chkconfig nfs on

chkconfig portmap on

SERVICIOS DEL NODO CABECERA

Claves SSH

Una parte importante del "pegamento" que mantiene unido y funcionando correctamente al cluster es el protocolo SSH.

Este es un protocolo (sobre IP) con múltiples funcionalidades, siendo la principal la ejecución de sesiones remotas en máquinas UNIX. Supone bastantes ventajas sobre el viejo "telnet", la más importante el cifrado de las comunicaciones.

El hecho de que las comunicaciones de las sesiones SSH sean cifradas (además de forma bastante efectiva) hace que muchas veces se conozca a estas sesiones como túneles. El concepto "túnel" es propio de la jerga de las Redes Privadas Virtuales (VPN).

SSH permite diferentes métodos de cifrado y autenticación. Las sesiones SSH desde una máquina cliente hacia un servidor suelen solicitar un nombre de usuario y una contraseña. Este es el modo más habitual de utilizarlo, pero una forma más efectiva en cuanto a seguridad es el uso de pares de claves. De forma similar a lo que ocurre con los cifrados PGP con clave pública, es posible autorizar a un determinado host cliente para abrir una sesión SSH en otra máquina siempre que disponga de la otra mitad de un par de claves pública-privada.

El uso de pares de claves presenta una gran ventaja para el objetivo de este trabajo. Habitualmente, los procesos de gestión de los nodos de un cluster no son ejecutados de forma interactiva por un operador humano, sino por scripts batch ejecutados en las máquinas cabecera o managers. Si esta tarea precisa ejecutar remotamente un comando en un nodo, es poco probable que el script sepa introducir un usuario y una contraseñas, esperar el prompt y ejecutar el comando. De igual forma, los anillos MPI se controlan mediante conexiones SSH entre los nodos que los componen. Estos anillos deben ser creados por las herramientas de ejecución de cada MPI y no tienen interacción humana.

Usando pares de claves, se consigue que un determinado usuario de una determinada máquina pueda hacer login en otra siempre que haya sido autorizada. La autorización consiste en disponer de una de las claves del par de cifrado.

En resumen, se busca que el usuario "root" del nodo manager pueda hacer login directamente a todos los nodos sin necesidad de introducir usuario y contraseña .

En SSH, los pasos para permitir que un determinado ***usuario cliente*** de una ***máquina cliente*** pueda hacer login directo en una ***máquina servidor*** con el mismo usuario son:

El usuario debe existir en ambas máquinas. En este caso el usuario 'root'

El usuario cliente generará un par de claves en la máquina cliente. El par de claves consiste en dos archivos, uno de ellos con extensión ".pub" llamado clave pública y otro con el mismo nombre pero sin la extensión ".pub", llamado clave privada.

Se copia el contenido del archivo de clave pública a un archivo especial de directorio HOME del usuario de la máquina servidora. Este es un archivo, habitualmente llamado "authorized_keys",

que contiene las claves públicas de todas las máquinas a las que se permite acceder sin usar usuario y password.

El usuario cliente, en la máquina cliente, intenta una conexión a la máquina servidora. Obtendrá directamente el *prompt* en ella.

Como el objetivo inicial es que el usuario root del manager pueda acceder como root a los nodos y al servidor de almacenamiento, el manager es la máquina cliente. En ella se debe generar el par de claves.

Los nodos y el servidor de almacenamiento son las máquinas servidoras y a todas ellas se deberá copiar el contenido de la clave pública generada en el manager.

Manager:

Asegurarse de ser el usuario “root”

 # whoami

Root

Generar el par de claves con cifrado RSA.

```
[root@localhost .ssh]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
b2:76:ce:9a:3c:0d:00:fb:f2:94:a2:50:45:59:76:bf root@manager.caligula
```

El comando “ssh-keygen” pide información sobre los nombres de los archivos de clave y su ubicación (aceptar por defecto). También solicita una contraseña con la que cifrar la propia clave. En este caso se deja en blanco y se continua pulsando ENTER.

El última línea que devuelve consiste en la huella o fingerprint. Esto es un hash que identificará el par de claves, indicando al final el usuario que la ha creado (root) y el nombre de la máquina en que se generó (manager.caligula).

Se puede echar un vistazo al contenido de los archivos de clave, que están almacenados en “/root/.ssh” con los nombres “id_rsa”(claves privada) e “id_rsa.pub” (clave pública).

```
[root@localhost .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtp/B/CrJ8IB6R7nzDj57C6uOybd4BleDr/3YF9Jvph6Psz4F1v+ARkCk7q3o1hKFd3gQ0V9MBVYo4vUs+y3fRY4gr
2udF4ORshuL5UwRbbX22H/6GnTANSJce6d+8u5WmTHATb9nZRELyz2sthcF+rrRRCSCKElxTrxTFhfiwYZncrG9kea22ZL3Wdftipx7g+nKrLqYUaxCmSe64Jil0e
NhdCFrdEC+QP8I11/fYAZMGq5RnjUkeBhbGvHJFrE64D76FlIXnzm3O5uD3ndjJ2vH7BUS4egD41v7+rycVd9uDF6MGCmVqG6AjmqunfD9FR5mh5XSrcXIIsZQXa5
```

```
[root@localhost .ssh]# cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAtp/B/CrJ8IB6R7nzDj57C6uOybd4BleDr/3YF9Jvph6Psz4F
1v+ARkCk7q3o1hKFd3gQ0V9MBVYo4vUs+y3fRY4gr2udF4ORshuL5UwRbbX22H/6
GnTANSJce6d+8u5WmTHATb9nZRELyz2sthcF+rrRRCSCKElxTrxTFhfiwYZncrG9
kea22ZL3Wdftipx7g+nKrLqYUaxCmSe64Jil0eNhdCFrdEC+QP8I11/fYAZMGq5R
njUkeBhbGvHJFrE64D76FlIXnzm3O5uD3ndjJ2vH7BUS4egD41v7+rycVd9uDF6M
GCmVqG6AjmqunfD9FR5mh5XSrcXIIsZQXa5LqQIBIwKCAQEApvh21+ybX4tofBe5
pp+GcRHhonvbc4M2kkcr+IXia/6vRMr+BmXxn1EL0uglVgJOJBYAv20STgWhuY+0
GNl7rU6+zEUh6Zzc+pzXrRJZE9mROviqJtEkwt2WYm1A3hu849W3IoMctCzXlT+z
2a6mYZTclhLAJNVu6OayXVeqv49IOdFe0SSpSX/M1SReyiKFHXLF/Wy4Aj8HyWNs
insVXisYeuIYrR0rjtuW1YxoQpjfDLdKIWIJKzcF7BwXsMupQCjHoMT8AGeth6wq
VMHO+JJD1stkkYBHPleJK2Iqr2LcedzYzEWFNfRrE5CATcaRGV/3kevjQCLjfQzs
Yug1CwKBgQDZiGehCmFAHkCSHSTnJQ/3U9rEKxJ0b9Onds8/ltqC352MDy+aREiF
bdeLlIa68/XveB0wWPnizCMDZ+MW6d2tWeFgNtBzHeYCjhDhgwexvf+XBpPHSAXR
DgPDs+l0zYamj+dmZAiL9OQ6ovtFpDGtBGQZK0QAC4PxLO1H5ODBuwKBgQDW6w0c
0sVdku6dM5M+uHbCl9l99Hk6n2OqnhO8kkeDk0aynoqGMqBRLudQSUeSYzkorKio
MLgHpLEpOM/4Q3TMaDD/h6ow4OJO4SZUDru3CYw2spLVmr3k6Sh6P1/I2ISmVy2Y
lLUAAIbwzglcpK4hXPlWlLfSFBuuJRLl7J+P6wKBgQDTUU65sk/X4t+j4cwTvZp7
O4QPBUUgpyVg2dCkLCS5pgbCkmjBvqzZY2OsKeH3cKXEDkg9mEM0FsLtXZrFzT2+
V0/3EK08oLONdBBmCkH9HvhJkVxbMAWmgqSSOb4284oeJWRyGAhNcY1ARo2xXayZ
cfrWnw7beOaSg2oolIIbRwKBgQDEfyHud4/ghlaBGTYqt0gCXvK8TTujT+YYVgNq
lF6kLuGNXcB6sfGpTRmqWYCLtUsgJozX74y37frLKDi+9iAX0K9vd1uhHcjjAXJ
Mgq1+hnLm/P2ftmA1SxSgxW+8dhdkYjGBE222+kWrb9qpTF9ltVWeVebgBlOws9z
IXvxSwKBgBh00eCuygYx0rSpqyY/M4L/StrEeg4l+eC1GCuGrYpQ6Yts4pQd5qwe
AGJlYCWDai9ltktNjS0yRUEPJBmF9KEcFIEzp5C+JTGpYDU63EE6Afe/9VVYeVEx
KfZAUaUt1xOolALO/gA59yUhUGmqDMWZ9pFHt8JYsSflQoDSbO0Z
-----END RSA PRIVATE KEY-----
```

Copiar la clave pública a los nodos.

El archivo que define que claves de cliente son aceptadas en un servidor se llama "authorized_keys" y hay uno de estos por cada usuario en el directorio oculto ".ssh" de su directorio HOME.

En el caso de los nodos, el archivo es: /root/.ssh/authorized_keys

En caso de que no esté creado, hay que crearlo y darle los permisos adecuados:

Nodo1# mkdir /root/.ssh

Nodo1# chmod 700 /root/.ssh

Nodo1# touch /root/.ssh/authorized_keys

El archivo "authorized_keys" podría contener otras claves. Cada línea será la clave pública de un usuario@host al que se permite acceder sin contraseña. En este caso, todos los nodos del cluster (excepto el manager), deben contener una línea en este archivo con la clave pública del manager. Se puede copiar el contenido "a mano" o por cualquier método que consiga este objetivo. También se puede utilizar la herramienta "ssh-copy-id" desde el manager, que hace todo el trabajo de crear el archivo y copiar la clave de forma remota a otras máquinas.

```
[root@caligula .ssh]# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtp/B/CrJ8IB6R7nzDj57C6uOybd4BleDr/3YF9Jvph6Psz4F1v+ARkCk7q3o1hKFd3gQ0V9MBVYo4vUs+y3fRY4gr
2udF4ORshuL5UwRbbX22H/6GnTANSJce6d+8u5WmTHATb9nZRELyz2sthcF+rrRRCSCKElxTrxTFhfiwYZncrG9kea22ZL3Wdftipx7g+nKrLqYUaxCmSe64Jil0e
NhdCFrdEC+QP3111/rAzHGq5RnjUkeBhbGvHJFrE64D76FlIXnzm3O5uD3ndjJ2vH7BUS4egD41v7+rycVd9uDF6MGCmVqG6AjmqunfD9FR5mh5XSrcXIIsZQXa5
LqQ== root@manager.caligula
```

Una vez hecho esto, el usuario ‘root’ del manager debe poder hacer login en cualquier nodo del cluster sin usar usuario/contraseña.

Puntos de montaje

Es necesario “montar” todos los recursos NFS que se han generado previamente en el nodo “storage”.

Como CentOS integra por defecto las herramientas para montar sistemas de archivos NFS, solo es necesario realizar los siguientes pasos (en **todos los nodos de ejecución y el manager**).

Crear el punto de montaje para ‘software’

 # mkdir /software

Montar los directorios ‘home’ y ‘software’:

 # mount –t nfs 10.9.9.12:/storage/home /home

 # mount –t nfs 10.9.9.12:/storage/software /software

Comprobar que el montaje ha sido correcto:

```
[root@caligula ~]# df -h
S.ficheros            Tamaño Usado  Disp Uso% Montado en
/dev/mapper/VolGroup00-LogVol00
                        6,7G  2,1G  4,4G  32% /
/dev/sda1               [illegible]
tmpfs                   [illegible]
/dev/hdc                [illegible]
10.9.9.12:/storage/home
                        6,7G  2,1G  4,4G  32% /home
10.9.9.12:/storage/software
                        6,7G  2,1G  4,4G  32% /software
```

Para asegurarse que en futuros reinicio estos recursos se montan automáticamente, añadir las siguientes líneas al archivo “/etc/fstab”

10.9.9.12:/storage/home /home nfs defaults

10.9.9.12:/storage/software /software nfs defaults

Ni que decir tiene que el recursos NFS debe ser **el primero en arrancar y el último en detenerse** en caso de querer reiniciar el cluster completo.

Servicio de hora

Para el buen funcionamiento del cluster, es fundamental que todas las máquinas que lo componen estén sincronizadas a nivel horario.

Las aplicaciones MPI utilizan la hora del sistema como marca de tiempo en la comunicación de mensajes del cálculo paralelo.

Existe un servicio basado en IP y modelo cliente servidor llamado NTP (Network Time Protocol). El servidor ofrece su hora a través de la red y los clientes se encargan de sincronizarse contra él de forma periódica (http://es.wikipedia.org/wiki/NTP).

Normalmente la hora del sistema (la que se ve en el sistema operativo de una máquina) proviene del reloj interno, físico, de la propia máquina. Pero los relojes de los ordenadores no son excesivamente precisos y tienden a retrasarse o adelantarse, lo cual hace que varias un conjunto de máquinas que estuvieran sincronizadas mediante su reloj interno, tengan horas diferentes después de un periodo de tiempo. Los desfases de unos pocos segundos entre nodos de un cluster MPI ya son significativas sobre el rendimiento del cálculo paralelo, sobre todo el programas con largos tiempos de proceso.

El protocolo NTP, simple en sus objetivos, es complejo a nivel interno. Los servidores de hora suelen ser, a la vez clientes de otros servidores de hora, formándose una estructura de árbol con los servidores que solo hacen de clientes en las hojas del mismo.

La jerarquía de servidores NTP se establece en “STRATUM’s”. Existen servidores:

- Stratum 0: Son los que disponen de la hora más fiable porque la obtienen de una fuente considerada “exacta”, como un reloj atómico. Obviamente, solo hay unos pocos servidores “Stratum 0” en el mundo
- Stratum 1: Aquellos que se sincronizan contra un servidor Stratum 0. En España se dispone de uno de estos servidores, perteneciente al Real Observatorio de la Armada y que toma la hora de un reloj atómico (http://es.wikipedia.org/wiki/Hora_ROA).
- Stratum 11: Aquellos que se sincronizan contra un servidor Stratum 1.
- Etc…
- Stratum 10: Reloj físico del sistema.

A menor nivel de Stratum de un servidor de hora, más correcta se considera la hora. Los clientes NTP suelen tener asociados varios servidores de hora contra los que sincronizarse. Obviamente siempre tratarán de sincronizarse contra el de menor Stratum posible.

Los servidores “Stratum 0” no son servidores realmente, sino que se consideran las fuentes de hora primarias, con desviaciones menores al segundo cada millón de años. Casi en su totalidad, provienen de relojes atómicos. El reloj atómico de la ROA es el encargado de dar la hora oficial en España.

Los servidores “Stratum 1” son los más conocidos en Internet. Son los que usan la hora de los “Stratum 0”. Aunque pueda parecer difícil, es relativamente fácil construir un servidor “Stratum 1”; esto es debido a que, en contra de lo que pueda parecer, es relativamente fácil tener acceso a la hora de un reloj atómico. Todos los satélites de la constelación GPS llevan a

bordo un reloj atómico de cesio y entre las señales de sincronización que envían a los receptores GPS en tierra, está la hora. De ahí que muchas instituciones cuenten con servidores de Stratum 1. Por ejemplo Rediris (hora.rediris.es), la FCSCL (hora.fcsc.es), el IAA, etc.

Como puede verse, el reloj físico del sistema se considera de los menos precisos en la cadena de Stratum y solo es útil como último recurso.

Para la sincronización de este cluster, se va a instalar:

- Un servidor NTP en el nodo manager (caligula.master), que se sincronizará contra el Stratum 0 de "hora.roa.es" (Real Observatorio de la Marina). Este va a ser un Stratum 1.
- Un cliente NTP en el resto de nodos, que se sincronizarán contra el nodo manager.

En la práctica, todo lo haremos con el paquete "ntpd", que permite hacer tanto de cliente como de servidor.

Instalación del servidor en el nodo manager (caléndula.cluster):

Instalar el paquete "ntpd" en caso de que no esté ya instalado, tanto en el nodo que va a hacer de servidor como en todos los clientes:

 # yum install ntp

Modificar el archivo de configuración del servidor para conseguir:

- Que actualice su hora contra el servidor "hora.roa.es".
- Que permita a los otros nodos del cluster, actualizarse contra él.

El archivo de configuración es "/etc/ntp.conf" y este es su contenido

```
[root@caligula ~]# cat /etc/ntp.conf | grep -v "#"
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict -6 ::1
restrict 10.9.9.0 mask 255.255.255.0 nomodify notrap noquery
server hora.roa.es
fudge   127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
```

Las líneas que se han modificado son:

server hora.fcsc.es

Establece contra que servidor se va a sincronizar el servicio. Normalmente se querrá poner al menos otra línea server más, que permita consultar un segundo servidor en caso de que el primero no esté disponible.

restrict 10.9.9.0 netmask 255.255.255.0

Permite que los clientes de la subred 10.9.9.0/24 puedan sincronizarse contra este servicio.

Arrancar el servicio

 # service ntpd start

Activar el inicio del servicio en el arranque del sistema:

 # chkconfig ntpd on

Comprobación de que el servicio está ejecutándose:

 # service ntpd status

 # ps aux | grep ntpd

```
[root@caligula ~]# service ntpd status
Se está ejecutando ntpd (pid  15695)...
[root@caligula ~]# ps aux | grep ntpd
root     10294  0.0  0.2   4044   724 pts/0    R+   18:56   0:00 grep ntpd
ntp      15695  0.0  1.3   4512  4508 ?        SLs  18:39   0:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
```

Ahora es momento de comprobar que el servidor se está sincronizando contra "hora.roa.es". Para ello se usará la herramienta "ntpq" con el parámetro "-p".

```
[root@caligula ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 hora.roa.es     .GPS.            1 u    3   64    1  366.810   46.713   0.061
 LOCAL(0)        .LOCL.          10 l    2   64    1    0.000    0.000   0.061
```

Con esto se puede ver una línea por cada fuente de tiempo contra la que el servidor intentará actualizarse. En este caso, la primera, como "Stratum 1 (columna st)" es "hora.roa.es" y la segunda el propio reloj interno de la máquina.

En la primera ejecución, recién arrancado el servicio NTP, muestra un desfase (columna delay) entre la hora del sistema y la del servidor de la ROA de varios minutos.

El protocolo no hace que la sincronización sea instantánea, sino que usa un algoritmo que va adelantando o retrasando poco a poco la hora del sistema para que tienda de forma asintótica a la hora del servidor. Por eso en la siguiente ejecución del comando "ntpq –p" unos segundos más tarde, se puede ver que la deriva ya es menor, e irá bajando hasta llegar a cero o casi cero.

```
[root@caligula ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 hora.roa.es     .GPS.            1 u    1   64    1  308.056    8.309   0.977
```

Configuración del resto de nodos del cluster

Para el resto de nodos del cluster, es necesario:

- Instalar el paquete ntp: *yum install ntp*
- Configurar el archivo "/etc/ntp.conf" con la línea "server 10.9.9.11"
- Arrancar el servicio*: service ntpd start*
- Activar el inicio del servicio en el arranque: *chkconfig ntpd on*
- Comprobar el funcionamiento con *"ntpq –p*"

```
[root@storage ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 10.9.9.11       LOCAL(0)        11 u   12   64    1    6.153   -7.031   0.122
 LOCAL(0)        .LOCL.          10 l   11   64    1    0.000    0.000   0.122
```

Nota: ntpdate. Algunos administradores de sistemas, optan por otra política a la hora de mantener un conjunto de máquinas sincronizadas. En lugar de instalar servicios "ntp" en todas ellas, usan de forma periódica el comando "ntpdate".

Al contrario que el servicio ntp, que está de forma continua haciendo tender la hora del sistema hacia la hora de un servidor ntp específico, "ntpdate" hace la sincronización completa de una vez. El problema es que debe ser ejecutado de forma periódica (vg.- cada 30 minutos) para corregir los constantes desvíos del sistema.

La elección de uno u otro procedimiento puede ser una cuestión de gustos, aunque se considera que el uso del servicio "ntp" de forma continua es más eficiente, ya que tiene la cualidad de *aprender* cual es la tendencia del reloj de sistema (si tiende a retrasarse o a adelantarse) y después de un tiempo es capaz de corregir la hora con muchas menos consultas al servidor.

Base de autenticación compartida. LDAP

Instalación del servidor Open-LDAP en CentOS (5.7)

Paquetes:

#> yum install openldap-servers openldap-clients

Configuración:

#> mkdir /var/lib/ldap/auth

#> chown ldap.ldap /var/lib/ldap/auth

#> chmod 700 /var/lib/ldap/auth

#> cp /etc/openldap/DB_CONFIG.example /var/lib/ldap/auth/DB_CONFIG

#> chown ldap.ldap /var/lib/ldap/auth/DB_CONFIG

#> chmod 600 /var/lib/ldap/auth/DB_CONFIG

Crear password del administrador de openldap

#> slappasswd

```
[root@localhost openldap]# slappasswd
New password:
Re-enter new password:
{SSHA}uzfDDqmyqI3xmdeXRrd04X7KdgouX7Nr
```

Editar «/etc/openldap/sldap.conf»

[...]

suffix *“dc=caligula,dc=net”*

rootdn *“cn=admin,dc=caligula,dc=net”*

[...]

rootpw *{SSHA}uzfDDqmyqI3xmdeXRrd04X7KdgouX7Nr*

[...]

directory */var/lib/ldap/auth*

[...]

#> /etc/init.d/ldap start

#> /etc/init.d/ldap restart

```
[root@localhost ldap]# /etc/init.d/ldap restart
Parando slapd:                                             [FALLÓ]
Verificando los archivos de configuración para slapd:  config file testing succ
eded
                                                           [  OK  ]
Iniciando slapd:                                           [  OK  ]
[root@localhost ldap]# /etc/init.d/ldap restart
Parando slapd:                                             [  OK  ]
Verificando los archivos de configuración para slapd:  config file testing succ
eded
                                                           [  OK  ]
Iniciando slapd:                                           [  OK  ]
```

#> chkconfig ldap on

Abrir puerto TCP:389 de entrada (no necesario si el firewall fue desactivado)

Comprobar que el servicio funciona

#> netstat -luntp | grep 389

#> ps aux | grep slap

#> telnet localhost 389

Comprobar que se «pobló» la base de datos de autenticatión de openldap

#> ls /var/lib/openldap/auth

(debe haber nuevos archivos con nombre: __db.00X

Instalar *phpldapadmin* para ver la base de datos;

La aplicación es web, de forma que hay que instalar un servidor web + php + php-ldap

#> yum install httpd php php-ldap

#> /etc/init.d/httpd start

#> chkconfig httpd on

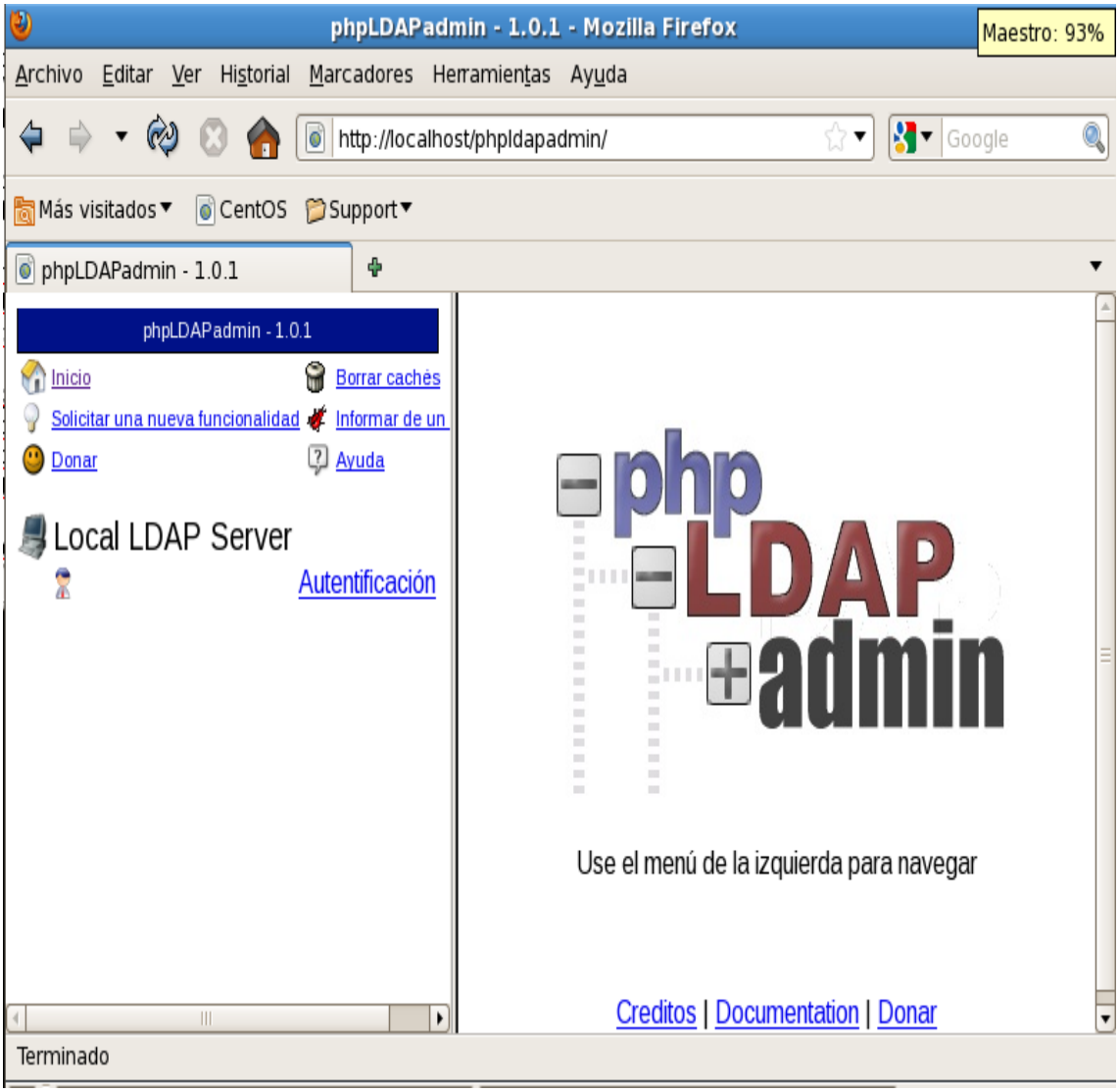
Descargar e instalar phpldapadmin desde rpm:

#> wget ...../phpldapadmin-1.0.1-1.el5.noarch.rpm

#> rpm -ivh phpldapadmin-1.0.1-1.el5.noarch.rpm

#> /etc/init.d/httpd restart

Comprobar que *phpdapadmin* funciona



Reconfigurar phpdapadmin para que permita conectarse al servidor ldap local (la configuración por defecto tiene un error):

/etc/phpldapadmin/config.php

```
/*********************************************/
/* Define your LDAP servers in this section  */
/*********************************************/

$i=0;
$ldapservers = new LDAPServers;
$ldapservers->SetValue($i,'server','name','My LDAP Server');
$ldapservers->SetValue($i,'server','host','127.0.0.1');
$ldapservers->SetValue($i,'server','port','389');
$ldapservers->SetValue($i,'server','auth_type','cookie');
$ldapservers->SetValue($i,'login','dn','cn=admin,dc=caligula,dc=net');
```

Conectarse y comprobar que la base de datos de usuarios está vacía.

Por defecto la base de datos está vacía. Es necesario hacer una migración de los usuarios locales (los que están en los archivos /etc/passwd y /etc/shadow) a LDAP. Para ello se usa como ayuda unas herramientas que provee el paquete OpenLDAP y que están disponibles en /usr/share/openldap/migration.

Las importaciones y exportaciones de datos en LDAP se realizan a través de ficheros de texto de extensión .ldif. En este caso, se usarán las herramientas de migración para generar los archivos ldif a partir de los passwd y shadow y después se importarán estos a la base de datos de OpenLDAP.

Pero previamente es necesario realizar un proceso previo, que consiste en crear la estructura básica del LDAP. LDAP sigue una estructura en árbol, con grupos y usuarios colgando unos de otros en diferentes Organizational Units.

Generación de la estructura LDAP:

Editar /usr/share/openldap/migration/migrate_common.ph

[...]

$DEFAULT_MAIL_DOMAIN = "caligula.net";

[...]

$DEFAULT_BASE = "dc=caligula,dc=net";

[...]

Generar archivo ldif con la estructura base:

#> perl /usr/share/openldap/migration/migrate_base.pl > /tmp/base.ldif

Comprobar el contenido del archivo:

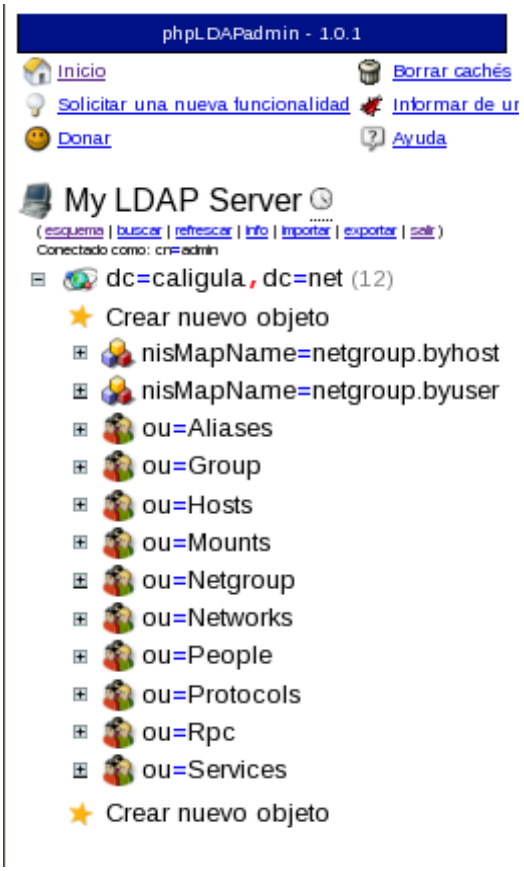
#> cat /tmp/base.ldif

Importar el archivo «base.ldif» a la LDAP:

#> ldapadd -a -x -W -D "cn=admin,dc=caligula,dc=net" -f /tm/base.ldif

```
[root@localhost migration]# ldapadd -a -x -W -D 'cn=admin,dc=caligula,dc=net' -f /tmp/base.ldif
Enter LDAP Password:
adding new entry "dc=caligula,dc=net"

adding new entry "ou=Hosts,dc=caligula,dc=net"

adding new entry "ou=Rpc,dc=caligula,dc=net"

adding new entry "ou=Services,dc=caligula,dc=net"

adding new entry "nisMapName=netgroup.byuser,dc=caligula,dc=net"

adding new entry "ou=Mounts,dc=caligula,dc=net"

adding new entry "ou=Networks,dc=caligula,dc=net"

adding new entry "ou=People,dc=caligula,dc=net"

adding new entry "ou=Group,dc=caligula,dc=net"

adding new entry "ou=Netgroup,dc=caligula,dc=net"

adding new entry "ou=Protocols,dc=caligula,dc=net"

adding new entry "ou=Aliases,dc=caligula,dc=net"

adding new entry "nisMapName=netgroup.byhost,dc=caligula,dc=net"
```

Comprobar que se han creado la OU en LDAP



Generar los ficheros ldap de usuarios y grupos e importarlos en LDAP:

#> perl migrate_group.pl /etc/group > /tmp/groups.ldif

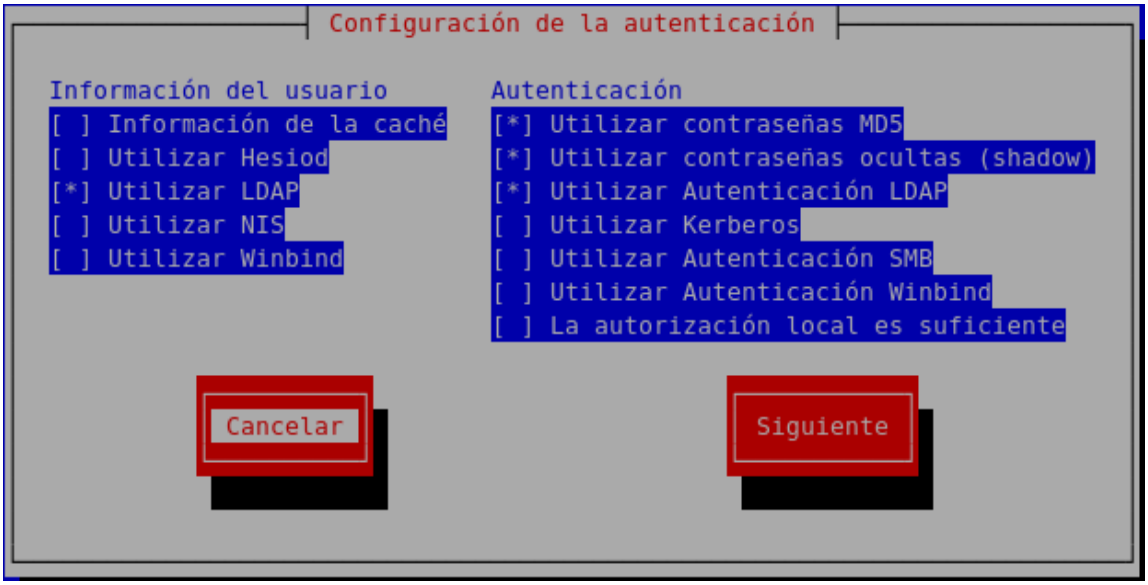
#> perl migrate_passwd.pl /etc/passwd > /tmp/users.ldif

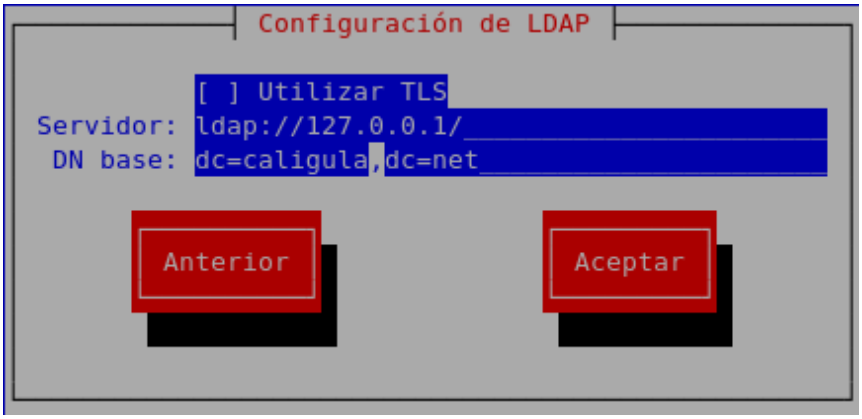
#> ldapadd -a -x -W -D "cn=admin,dc=caligula,dc=net" -f /tmp/groups.ldif

#> ldapadd -a -x -W -D "cn=admin,dc=caligula,dc=net" -f /tmp/users.ldif

Modificar todos los nodos para autentificar desde LDAP

#> authconfig-tui





Comprobar que la autentificación centralizada funciona.

Añadir nuevos usuarios a la base de datos LDAP:

La instalación se ha realizado con el usuario *root* y aunque su uso es esencial para la administración del sistema, no es aconsejable utilizarlo para la operación diaria, el desarrollo de programas y la ejecución de los mismos.

La creación de nuevos usuarios implica generar sus cuentas en la base de datos LDAP y los directorios HOME en el almacenamiento NFS.

Además, es necesario generar un par de claves SSH para los nuevos usuarios que permitan iniciar sesiones remotas en los nodos de cálculo. Esto es necesario para que los usuarios no privilegiados puedan ejecutar programas MPI que precisan de anillos MPI controlados por SSH (como en OpenMPI).

El proceso es el siguiente:

Generar las cuentas de grupo y usuario en LDAP

Se pueden utilizar archivos de texto '.ldif' importados o gestores gráficos como PHPLdapAdmin o LDAPBrowser. En este caso se hará con un archivos '.ldif'.

Primero se deben crear un par de archivos (aunque podría hacerse solo con uno) con la información del nuevo usuario y su grupo. Los archivos son de texto y con extensión ".ldif".

Nuevo_grupo.ldif

dn: cn=<nombre_grupo>,ou=group,dc=caligula,dc=net
changetype: add
objectClass: posixGroup
objectClass: top
cn: <nombre_grupo>
gidNumber: 1000
memberUid:<nombre_usuario>

Nuevo_usuario.ldif

```
dn: cn=<nombre_usuario>,ou=people,dc=caligula,dc=net
objectclass: inetOrgPerson
cn: <nombre_usuario>
uid: <nombre_usuario>
userpassword: <password>
```

Cargar los archivos '.ldif' en la base de datos ldap:

#> ldapadd -a -x -W -D "cn=admin,dc=caligula,dc=net" -f Nuevo_grupo.ldif

#> ldapadd -a -x -W -D "cn=admin,dc=caligula,dc=net" -f Nuevo_usuario.ldif

Crear el directorio HOME del usuario

 # mkdir /home/nombre_usuario

 # chown nombre_usuario:nombre_grupo /home/nombre_usuario

 # chmod 750 /home/nombre_usuario

(Generar un archivo "$HOME/.bashrc" con la configuración de sesión y las variables de entorno necesarias)

Generar un par de claves SSH y permitir sesiones remotas en los nodos de cálculo.

 # su – nombre_usuario (o haciendo login previo como el usuario).

 # ssh-keygen –t rsa

 # cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

Ahora se debe probar que el usuario puede hacer login con su cuenta en los nodos de cálculo. Como el almacenamiento donde está el directorio raíz del usuario está compartido por NFS y es físicamente el mismo en todos los nodos, no es necesario copiarlo a cada uno de ellos.

 # su – nombre_usuario

 # ssh nodo1

Nodo1#

También debería ser capaz de ejecutar programas MPI con el comando “mpirun” y los parámetros habituales para ejecutarse en múltiples nodos.

Todos estos pasos pueden hacerse de forma sencilla usando gestores de cuentas. En este caso se explica, de forma intencionada, el caso menos cómodo para fomentar el aprendizaje del lector.

COMPILACIÓN E INSTALACIÓN DE LIBRERÍAS MPI DE OPENMPI

OpenMPI, como todas las librerías MPI consiste en una serie de bibliotecas de programación que se pueden utilizar llamar desde diferentes compiladores y diferentes lenguajes de programación.

El HPC, los más habitual es la programación en lenguajes C o Fortran. Los compiladores más utilizados suelen ser GCC (GNU C Compiler, de código abierto) e ICC (Intel C Compiler, propietario). Tanto GCC como ICC permiten compilar código fuente en otros lenguajes de programación diferentes a C. Dentro de las implementaciones de librerías MPI también hay opciones open source (MPICH2, OpenMPI, LAMP) y propietarias (Intel MPI, Platform MPI). En cualquier caso y como si se tratara de cualquier otra librería de programación se distribuye:

- Una parte en código fuente (.h), necesario para compilar programas que hagan llamadas a estos.

- Una parte binaria, precompilada formada por archivos objeto para enlazar programas MPI. También algunas herramientas de compilación y depuración. Además, los programas MPI suelen necesitar de ciertas herramientas en tiempo de ejecución para funcionar correctamente.

La instalación de una distribución MPI no se diferencia de la de otros softwares de desarrollo.

Desde el nodo manager:

Instalar las herramietnas de compilación de CentOS:

#> yum groupinstall “Development tools”

Descargar la versión de OpenMPI elegida. En este caso la 1.4.4, última estable:

#> wget http://www.open-mpi.org/software/ompi/v1.4/downloads/openmpi-1.4.4.tar.bz2

Compilar e instalar:

#> tar xvfz openmpi-1.4.4.tar.bz2

#> cd openmpi-1.4.4

#> ./configure --prefix=/software/openmpi144 --with-sge

#> make all

#> make install

NOTA: Asegurarse que todos los pasos se completan sin errores y que la ruta "/software/openmpi144" se llena de contenido.

Como se puede ver, se usa el esquema tradicional de compilación/instalación para paquetes de software fuente (configura, make, make install).

La única diferencia está en los parámetros utilizados en "./configure". Se usan los siguientes:

«--prefix=<PATH>» Con este parámetro se consigue modificar la ruta a la que irán a parar los archivos instalados. Se utilizará una ruta de disco compartida en todos los nodos de ejecución y el nodo cabecera. De esta forma todos los programas MPI tendrán disponibles todos los recursos MPI necesarios sin necesidad de tener que ir instalando uno a uno todos los nodos.

«--with-sge»: SGE (Sun Grid Engine) u OGE (Oracle Grid Engine) es un software de gestión de colas de trabajo. En el caso de que los programas MPI tengan que funcionar a través de un gestor de colas SGE, con esta opción se consigue una mejor integración entre ambos.

Con estos pasos, todas las librerías MPI de OpenMPI, además de algunos paquetes binarios de ayuda en la compilación y ejecución de programas MPI, están disponibles desde cualquier lugar del cluster en la misma ruta "/software/openmpi143".

TESTEANDO EL ENTORNO MPI. LANZAMIENTO DEL PRIMER PROGRAMA PARALELO MPI «HELLO WORLD»

A continuación se detallan los pasos para escribir un programa MPI en C, compilarlo contra las librerías MPI de OpenMPI instaladas previamente y ejecutarlo en varios hilos de forma que cada uno se ejecute en un nodo diferente.

El **código fuente** del programa, llamado «hello_world_mpi.c» es:

```
#include <stdio.h>
#include <mpi.h>

int main (argc, argv)
        int argc;
        char *argv[];
{
        int rank, size;
        char name[BUFSIZ];
        int length;

        MPI_Init (&argc, &argv);
        MPI_Comm_rank (MPI_COMM_WORLD, &rank);
        MPI_Get_processor_name(name, &length);
        MPI_Comm_size (MPI_COMM_WORLD, &size);
        printf ("Hello world from process %d of %d in node:%s\n", rank, size, name);
        MPI_Finalize();
        return 0;
}
```

Compilación:

```
$ export LD_LIBRARY_PATH=/software/openmpi143/lib/

$ export PATH=$PATH:/software/openmpi143/bin

$ mpicc –o hello_world_mpi hello_world_mpi.c
```

Ejecución:

Muchos programas compilados en MPI pueden ejecutarse con una simple orden «./ejecutable», pero en el mejor de los casos, lo que se consigue con esto es que el programa se ejecute en un solo hilo, un solo nodo o un solo procesador. Es decir, sin paralelismo.

Para ejecutar programas MPI en paralelo, se hace uso de algunas herramientas propias del entorno MPI. Estas varían de unas distribuciones de MPI a otras aunque suelen ser similares.

La herramienta principal para la ejecución de programas paralelos MPI es «mpirun». Este comando, precediendo al archivo MPI ejecutable se encarga de hacer que el programa inicie diferentes hilos de ejecución y estos migren su ejecución a diferentes nodos y procesadores. Además, se encarga de mantener la comunicación entre todos los nodos que forman parte del proceso e informar al usuario del final del trabajo, posibles errores, etc.

En OpenMPI, todo este proceso se conoce como «generar el anillo MPI». Al ejecutar un «mpirun», se generan los siguientes eventos:

- El nodo que donde se ejecuta el «mpirun» recibe como parámetro el número de hilos que desea ejecutar el programa mpi y una lista con los hostnames de los nodos en los que desea ejecutarse. Obviamente estos dos parámetros deben ser consistentes entre ellos, ya que lo óptimo es solo ejecutar un hilo en cada core de cpu.

- El nodo decide, según algún criterio, en que nodos y cores de los disponibles se va a ejecutar el programa. Estos conforman el anillo MPI.

- Los nodos que formarán el anillo deben comunicarse entre sí para poder recibir sus hilos de proceso y saber que otros nodos participan en el trabajo. Esto se hace mediante conexiones, usualmente IP, de algún tipo entre todos ellos. En el caso de OpenMPI, el nodo donde se lanza el «mpirun» hace conexiones SSH a los demás nodos, levantando un pequeño proceso en cada uno específico para ese trabajo.

- En el momento que todos los nodos saben que otros nodos conformarán el anillo, se die que el anillo está creado. El nodo donde se lanza el «mpirun» hace las veces de manager para el trabajo. Y entre sus funciones está controlar el estado del trabajo en todo momento. Dependiendo de las necesidades del operador, se puede hacer que el nodo manager forme parte del anillo pero no ejecute ningún proceso MPI. Esto se hace para no sobrecargarlo.

En este caso de estudio, el nodo manager siempre es el mismo y ya que hace otras muchas tareas (gestión de usuarios, monitorización, etc), forma parte y controla todos los anillos MPI pero no ejecuta hilos MPI de ninguno de ellos.

- El nodo manager despliega los hilos de ejecución y controla su estado hasta el final. Una vez el trabajo finaliza (sea cual sea su estado), el anillo se destruye.

Viendo esto se puede comprender fácilmente que los programas deban estar en una ubicación compartida y visible por todos los nodos.

Puede haber diferentes anillos MPI funcionando a la vez y cualquier nodo puede estar a la vez formando parte de diferentes anillos MPI aunque no es recomendable. Si un nodo y sus recursos forman parte de dos anillos MPI diferentes, con toda seguridad estará ejecutando hilos de trabajos MPI diferentes. Estos competirán por los recursos y el rendimiento se verá penalizado.

En clusters pequeños en los que un solo operador lanza trabajos, es sencillo controlar cuantos anillos MPI hay y coordinar el lanzamiento de trabajos de forma secuencial para impedir que compitan entre ellos por los mismos recursos. En grandes clusters, como Caléndula, se evita que los usuarios puedan lanzar anillos MPI de forma autónoma, ya que al haber docenas de ellos a la vez, sería dificil ponerlos de acuerdo. En estos casos se utiliza un software intermedio llamado «gestor de colas» o «gestor de trabajos». Este software es el único que tiene el poder de crear anillos MPI y ejecutar trabajo paralelos. Los usuarios solamente pueden solicitar al gestor de colas la ejecución de su trabajo paralelo, que normalmente quedará encolado a la espera de que los recursos necesarios para su ejecución estén libres.

Lanzar la ejecución del programa MPI anterior consiste en las siguientes órdenes:

Creación de un archivo «machines», con los hostnames de los nodos que pueden participar en el trabajo:

#> echo manager >> ./machines

#> echo nodo1 >> ./machines

#> echo nodo2 >> ./machines

Definición de las variables de entorno necesarias para que el entorno de ejecución MPI esté disponible a la sesión.

#> export LD_LIBRARY_PATH=/software/openmpi143/lib/

#> export PATH=$PATH:/software/openmpi143/bin

Ejecución del comando mpirun:

#> mpirun -display-allocation -display-devel-map -nolocal -np 20 -machinefile machines hello_world

Explicación de los parámetros:

mpirun: Llamada al entorno de ejecución de programas MPI

-display-allocation: Muestra que nodos se han elegido para la ejecución

-display-devel-map: Similar al anterior pero ofrece información complementaria

-nolocal: Evita que el nodo master del anillo ejecute hilos MPI

-np 20: Número de hilos que ejecutará el programa

- machinefile machines:Nombre del archivo donde se definen las máquinas que pueden participar en el trabajo.

-hello_world: Ejecutable en MPI

La salida de este programa debe ser algo parecido a lo siguiente:

==================== ALLOCATED NODES ====================

Data for node: Name: caligula Num slots: 1 Max slots: 0

===

==================== ALLOCATED NODES ====================

Data for node: Name: caligula Launch id: -1 Arch: ffca0200 State: 2

Num boards: 1 Num sockets/board: 1 Num cores/socket: 1

Daemon: [[16806,0],0] Daemon launched: True

Num slots: 1 Slots in use: 0

Num slots allocated: 1 Max slots: 0

Username on node: NULL

Num procs: 0 Next node_rank: 0

Data for node: Name: nodo1 Launch id: -1 Arch: 0 State: 0

Num boards: 1 Num sockets/board: 1 Num cores/socket: 1

Daemon: Not defined Daemon launched: False

Num slots: 1 Slots in use: 0

Num slots allocated: 1 Max slots: 0

Username on node: NULL

Num procs: 0 Next node_rank: 0

Data for node: Name: nodo2 Launch id: -1 Arch: 0 State: 0

Num boards: 1 Num sockets/board: 1 Num cores/socket: 1

Daemon: Not defined Daemon launched: False

Num slots: 1 Slots in use: 0

Num slots allocated: 1 Max slots: 0

Username on node: NULL

Num procs: 0 Next node_rank: 0

==========

Map generated by mapping policy: 2400

Npernode: 0 Oversubscribe allowed: TRUECPU Lists: FALSE

Num new daemons: 2 New daemon starting vpid 1

Num nodes: 2

Data for node: Name: nodo1 Launch id: -1 Arch: 0 State: 0

Num boards: 1 Num sockets/board: 1 Num cores/socket: 1

Daemon: [[16806,0],1] Daemon launched: False

Num slots: 1 Slots in use: 10

Num slots allocated: 1 Max slots: 0

Username on node: NULL

Num procs: 10 Next node_rank: 10

Data for proc: [[16806,1],0]

Pid: 0 Local rank: 0 Node rank: 0

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],2]

Pid: 0 Local rank: 1 Node rank: 1

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],4]

Pid: 0 Local rank: 2 Node rank: 2

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],6]

Pid: 0 Local rank: 3 Node rank: 3

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],8]

Pid: 0 Local rank: 4 Node rank: 4

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],10]

Pid: 0 Local rank: 5 Node rank: 5

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],12]

Pid: 0 Local rank: 6 Node rank: 6

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],14]

Pid: 0 Local rank: 7 Node rank: 7

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],16]

Pid: 0 Local rank: 8 Node rank: 8

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],18]

Pid: 0 Local rank: 9 Node rank: 9

State: 0 App_context: 0 Slot list: NULL

Data for node: Name: nodo2 Launch id: -1 Arch: 0 State: 0

Num boards: 1 Num sockets/board: 1 Num cores/socket: 1

Daemon: [[16806,0],2] Daemon launched: False

Num slots: 1 Slots in use: 10

Num slots allocated: 1 Max slots: 0

Username on node: NULL

Num procs: 10 Next node_rank: 10

Data for proc: [[16806,1],1]

Pid: 0 Local rank: 0 Node rank: 0

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],3]

Pid: 0 Local rank: 1 Node rank: 1

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],5]

Pid: 0 Local rank: 2 Node rank: 2

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],7]

Pid: 0 Local rank: 3 Node rank: 3

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],9]

Pid: 0 Local rank: 4 Node rank: 4

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],11]

Pid: 0 Local rank: 5 Node rank: 5

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],13]

Pid: 0 Local rank: 6 Node rank: 6

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],15]

Pid: 0 Local rank: 7 Node rank: 7

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],17]

Pid: 0 Local rank: 8 Node rank: 8

State: 0 App_context: 0 Slot list: NULL

Data for proc: [[16806,1],19]

Pid: 0 Local rank: 9 Node rank: 9

State: 0 App_context: 0 Slot list: NULL

Hello world from process 1 of 20 in node:nodo2

Hello world from process 2 of 20 in node:nodo1

Hello world from process 4 of 20 in node:nodo1

Hello world from process 3 of 20 in node:nodo2

Hello world from process 7 of 20 in node:nodo2

Hello world from process 11 of 20 in node:nodo2

Hello world from process 8 of 20 in node:nodo1

Hello world from process 5 of 20 in node:nodo2

Hello world from process 10 of 20 in node:nodo1

Hello world from process 9 of 20 in node:nodo2

Hello world from process 0 of 20 in node:nodo1

Hello world from process 6 of 20 in node:nodo1

Hello world from process 12 of 20 in node:nodo1

Hello world from process 13 of 20 in node:nodo2

Hello world from process 14 of 20 in node:nodo1

Hello world from process 18 of 20 in node:nodo1

Hello world from process 15 of 20 in node:nodo2

Hello world from process 17 of 20 in node:nodo2

Hello world from process 19 of 20 in node:nodo2

Hello world from process 16 of 20 in node:nodo1

Las últimas 20 líneas se corresponden con la salida del programa MPI. El resto corresponde a información de depuración MPI. Se muestran las máquinas que forman el anillo y como se distribuyen los procesos por ellas.

Este es un programa muy simple que muestra como los procesos se ejecutan en diferentes cores de diferentes máquinas y muestran información de donde están. Se puede ver como los 20 hilos se han repartido por los nodos del anillo. El anillo lo ha creado el nodo manager, que aun formando parte del mismo no ha ejecutado hilos MPI.

La compilación y ejecución de otros programas MPI es similar en todos los casos.

AÑADIR NUEVOS NODOS AL CLUSTER

Una de las características de los clusters es lo fácilmente ampliables que resultan. Como si fuera un juego de construcción, es posible añadir nuevos recursos a medida que se necesitas. Además, como estos recursos son componentes informáticos de consumo y baratos, supone una ventaja sobre las grandes máquinas específicas de cálculo de IBM, HP o SGI.

La ampliación de un cluster consiste normalmente en añadir nuevos nodos al mismo.

En la práctica añadir nuevos nodos a un cluster supone, normalmente, romper la homogeneidad del mismo. Aunque la única condición que se necesita es que los nuevos nodos tengan la misma arquitectura interna (en este caso Intel X86 32bits) y el mismo hardware de conectividad, normalmente un nuevo nodo tendrá diferencias respecto a los existentes. Es posible que el nuevo nodo sea idéntico (el caso óptimo), pero lo más probable es que disponga de otra cantidad o tecnología de memoria, otra familia de procesador dentro de la misma arquitectura, diferente velocidad de bus, etc.

Estos matices deben ser tenidos en cuenta. Aunque los clusters tienden a ser por definición sistemas altamente heterogéneos, añadir nuevos nodos con notables diferencias con los anteriores puede suponer problemas. El más común es la pérdida global de rendimiento del sistema (quizá lo contrario que se deseaba con los nuevos nodos).

Un cluster es similar a una cadena en tanto en cuanto ambos son tan fuertes como el más débil de sus eslabones. Se recomienda una estudio específico a la hora de añadir nodos de diferentes características a los existentes.

En este caso se añaden nuevos nodos que son idénticos a los existentes. El procedimiento es:

- Instalar el nuevo nodo con el mismo sistema operativo y versión que los anteriores. Pequeñas variaciones en las versiones pueden provocar errores inesperados. Normalmente no hay problemas si diferentes nodos tienen diferentes versiones de las “minor release” de los paquetes.
- Decidir su direccionamiento de red, compatible con las topología física (Ethernet) y lógica (direccionamiento) existente y configurarlo para que haga referencia al mismo.
- Instalar el servicio ssh en el nodo.
- Añadir la entrada correspondiente del nodo “manager” al archivo “/root/.ssh/authorized_keys” para que este pueda iniciar sesiones sin necesidad de contraseña.
- Crear los puntos de montaje “/software” y “/home”, montar los sistemas de ficheros NFS de la máquina storage y asegurarse de añadir las líneas necesarias en “/etc/fstab”.
- Configurar la autenticación del nodo para que permita usar el servicio LDAP del cluster.

- Añadir una nueva línea al archivo “/etc/hosts” de todos los nodos que refleje el hostname y la dirección IP del nuevo nodo. El contenido de este archivo también debe aparecer en el nuevo nodo.
- Añadir el nodo a los sistemas de monitorización existentes (nagios, ganglia…)
- Realizar una prueba de ejecución MPI. Lanzar un comando “mpirun” desde el manager con algún programa MPI (vg.- hello_world_mpi) pasando como parámetro un fichero “machines” que incluya al nuevo nodo.
- Comprobar que las ejecuciones son correctas.
- Muy importante. Comprobar la estabilidad del sistema y la ganancia o pérdida de rendimiento. Normalmente, después de una ampliación se realiza un nuevo proceso de caracterización del cluster.

SISTEMA DE MONITORIZACIÓN

Ganglia

Consta de dos partes:

- Un servidor central (en este caso en manager), donde corre el demonio «gmetad», encargado de recibir los datos de monitorización de los nodos del cluster y el demonio «ganglia-web», encargado de mostrar las estadísticas de forma gráfica apoyándose en un servidor web + php.

- Los nodos corren el demonio «gmond», encargado de recolectar las estadísticas de uso de recursos del nodo y enviarlas al demonio «gmetad» remoto.

En esta instalación, el servidor central de monitorización, donde se ejecutará el demonio «gmetad», estará instalado en el nodo manager. Los nodos de ejecución correrán cada uno el demonio «gmond». Las estadísticas se visualizarán a través del servidor web del nodo manager.

Instalación del servicio «gmetad», «ganglia-web» en nodo manager:

Instalar dependencias:

#> yum install php-gd

#> service httpd restart

Descargar los siguientes paquetes (http://rpmbone.net):

- rrdtool-1.2.19-1.el5.kb.i386.rpm

- ganglia-3.0.7-1.el5.kb.i386.rpm

- ganglia-gmetad-3.0.7-1.el5.kb.i386.rpm

- ganglia-gmond-3.0.7-1.el5.kb.i386.rpm

- ganglia-web-3.0.7-1.el5.kb.i386.rpm

Instalarlos:

#> rpm -ivh ganglia rrdtool**

#> servide httpd restart

Nota: Se instala el demonio «gmond» también en el nodo manager para que se monitoricen los recursos del mismo.

Configuración de «gmetad»:

/etc/ganglia/gmetad.conf

[...]

data_source "CALIGULA" 10.9.9.1:8649

[...]

data_source NOMBRE_CLUSTER IP_GMETAD:PORT

Arrancar los demonios

#> service gmond start

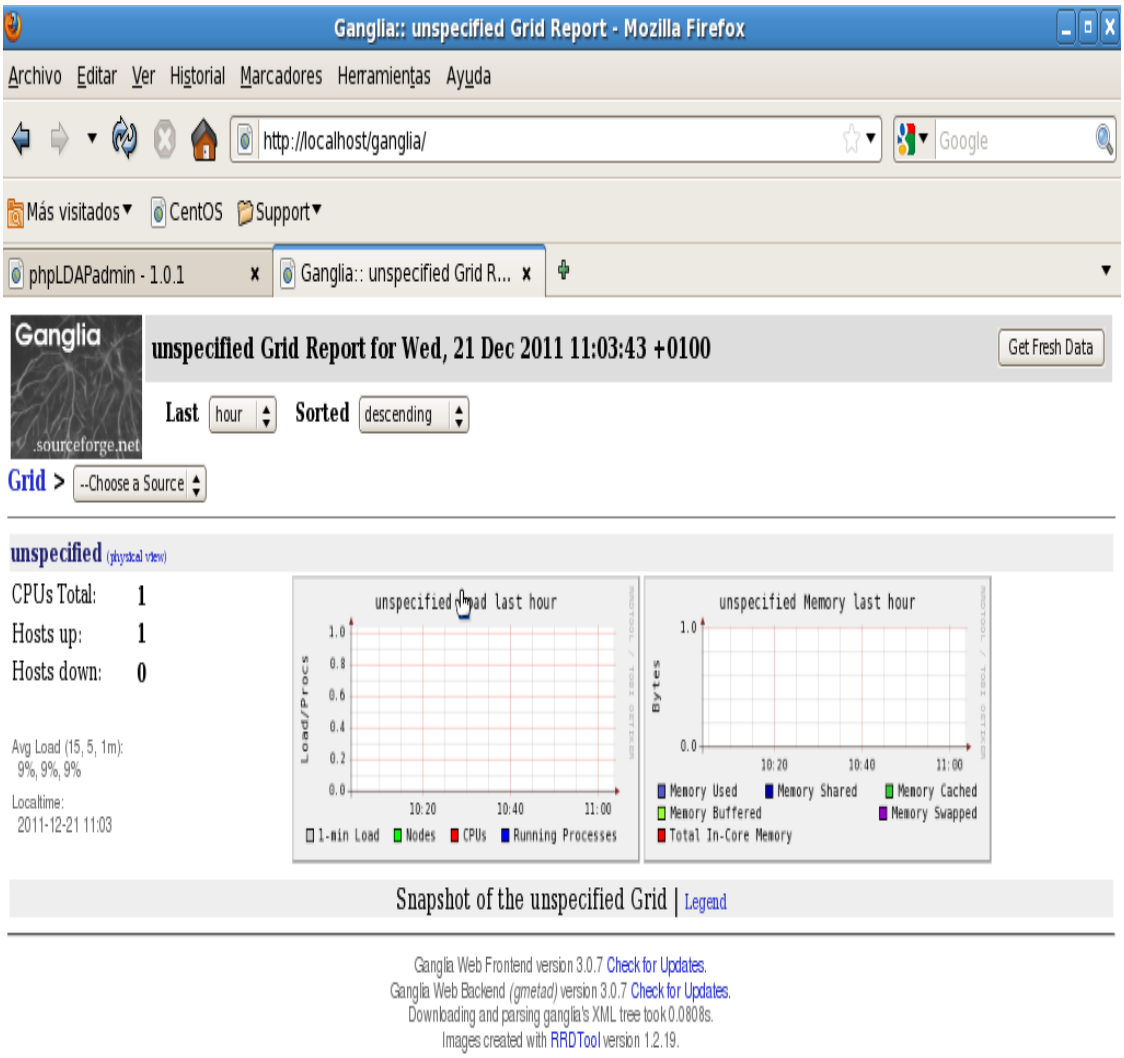
#> service gmetad start

#> chkconfig gmond on

#> chkconfig gmetad on

Ganglia funciona por UDP Multicast, de forma que los demonio «gmond» envían las estadísticas por multicast a toda la subred a la que estén conectados. En clusters mayores, sería recomendable eliminar este tipo de conexión y enviar las estadísticas vía TCP ó UDP solamente al nodo que ejecuta el «gmetad». De hecho, para los demonio «gmond» de los nodos se hará de esta manera.

Probar que las estadísticas se visualizan (aunque solo las del propio nodo manager).



Instalar el demonio «gmond» en los nodos

Hacer login en el nodo

Instalar los paquetes «ganglia-gmond-3.0.7-1.el5.kb.i386.rpm» « ganglia-3.0.7-1.el5.kb.i386.rpm»

#> rpm -ivh ganglia-gmond-3.0.7-1.el5.kb.i386.rpm

#> rpm -ivh ganglia-3.0.7-1.el5.kb.i386.rpm

Configurar el demonio a través del archivo «/etc/gmond.conf»

[...]

```
        cluster {
        name = "CALIGULA"
        [...]
        }
[...]
        udp_send_channel {
                #mcast_join = 239.2.11.71    #Deshabilitar multicast
                host = 10.9.9.11          #IP del servidor donde corre GMETAD
                port = 8649
                ttl = 1
        }
[...]
        udp_recv_channel {
                #mcast_join = 239.2.11.71       #Deshabilitado
                port = 8649
                #bind = 239.2.11.71             #Deshabilitado
        }
[...]
```

Arrancar el servicio

```
#> service gmond start
#> chkconfig gmond on
```

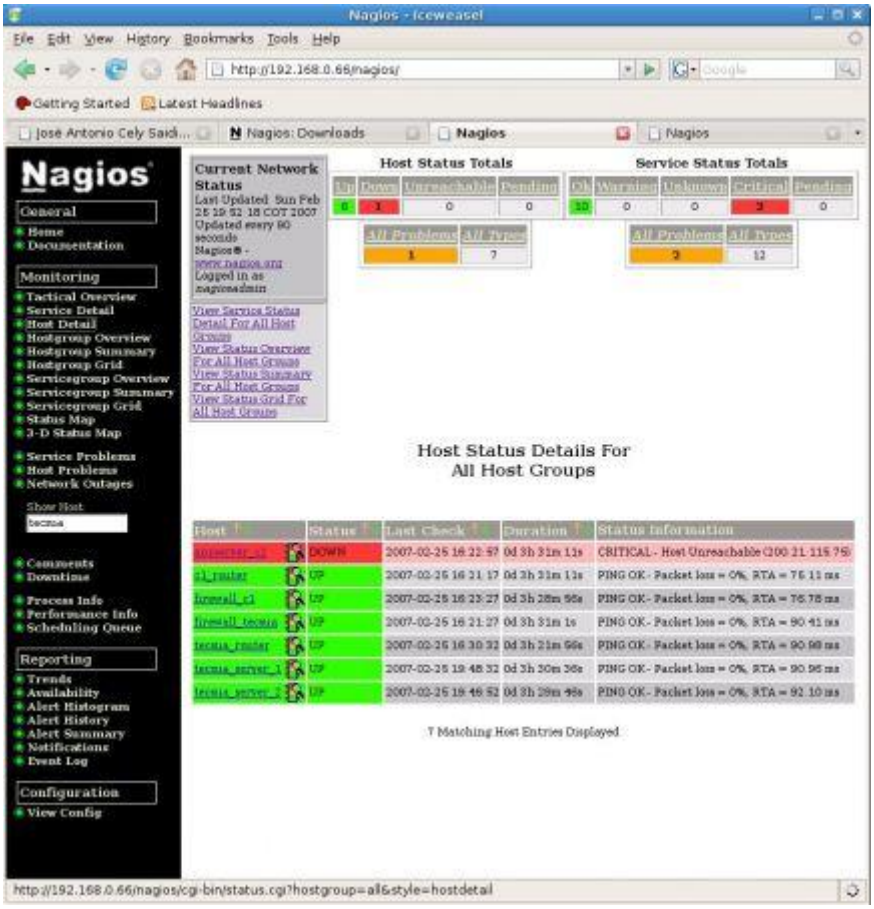
Comprobar que tras un tiempo, ganglia muestra las estadísticas relativas a este nodo.

Nagios

Nagios es una pieza de software libre muy conocida entre administradores de sistemas.

Es un software para monitorización de servidores y servicios. Se administra vía sus archivos de configuración y ofrece información gráfica (colores, iconos, mapas, etc…) sobre el estado de los "ítems" que monitoriza.

El un sistema muy escalable, de forma que sirve para monitorizar desde unos pocos nodos y servicios hasta cientos de miles.



Nagios es una opción muy interesante para monitorizar el cluster, de forma autónoma o como complemento a Ganglia. A continuación se explica, de forma muy somera, el proceso de instalación de Nagios para monitorizar el cluster. Queda a elección del lector ampliar su funcionalidad tanto como desee.

Se puede instalar Nagios en cualquiera de las máquinas del cluster, aunque lo más lógico sería instalarlo en el "manager".

Instalación de Nagios3 en "CentOS 5.7"

Nagios está formado por varios módulos que funcionan cooperativamente. Uno de ellos es el encargado de mostrar la información vía web. Para que funcione, se **necesita un servidor web con soporte PHP.**

Instalación: Apache + PHP

yum install httpd php php-gd

chkconfig httpd on

service httpd start

Otras dependencias necesarias para nagios

yum install gd-devel

yum install gcc

Como Nagios no está incluido en los repositorios oficiales de la distribución, será necesario compilarlo desde los fuentes.

Descargar los siguientes paquetes de la web oficial de Nagios:

Paquetes NAGIOS a descargar

nagios-3.3.1.tar.gz

nagios-plugins-1.4.15.tar.gz

Compilar/Instalar Nagios

useradd -m nagios

groupadd nagcmd

usermod -a -G nagcmd nagios

usermod -a -G nagcmd apache

tar xvfz nagios-3.3.1.tar.gz

cd nagios

./configure --prefix=/opt/nagios --with-nagios-user=nagios --with-nagios-group=nagcmd --with-comand-user=nagios --with-comand-group=nagcmd

make all

make install

make install-init

make install-config

make install-commandmode

make install-webconf

htpasswd -c /opt/nagios/etc/htpasswd.users nagiosadmin

sevice httpd restart

Una vez instalado Nagios, es necesario instalar sus plugins. Los plugins son pequeños programas/scripts que se encargar de monitorizar cosas muy concretas (vg.- Si un servidor web responde o si un disco ha superado el 90% de espacio ocupado). Los scripts devuelven los valors que monitorizan al nucleo de nagios que se encarga de mostrarlos al usuario, hacer estadísticas, etc.

Compilar/Instalar Nagios-plugins

tar xvfz nagios-plugins-1.5.15.tar.gz

cd nagios-plugins

./configure --with-nagios-user=nagios --with-nagios-group=nagios --prefix=/opt/nagios/

make

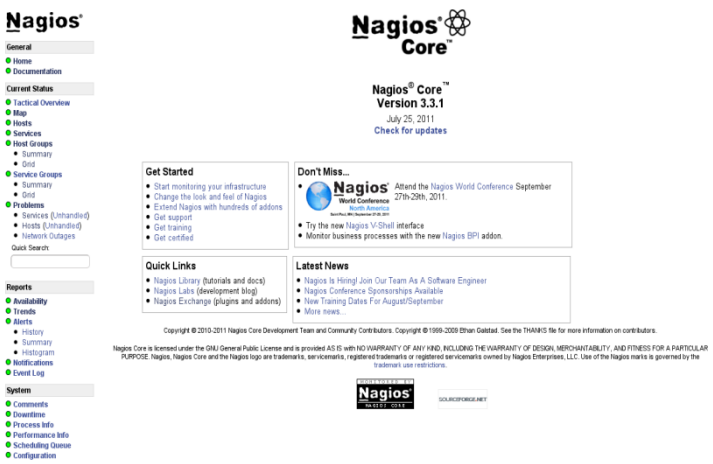
make install

Ya solo queda iniciar servicio nagios y comprobar que está operativo

service nagios restart

chkconfig nagios on

Comprobar: «http://xxx.x.x/nagios



En cuanto arranca, nagios ya tiene un host al que monitorizar; a sí mismo. Si se selecciona "Hosts" en el menú izquierdo se verá que existe un ítem "localhost", en verde, que indica que la máquina donde está instalado Nagios (el manager). En realidad, no tiene sentido monitorizar un hosts desde sí mismo, ya que si llegara a "caer", no se podría ver el estado de alarma del mismo ya que está caído.

Pero lo interesante es que sirve de ejemplo para añadir nuevos nodos.

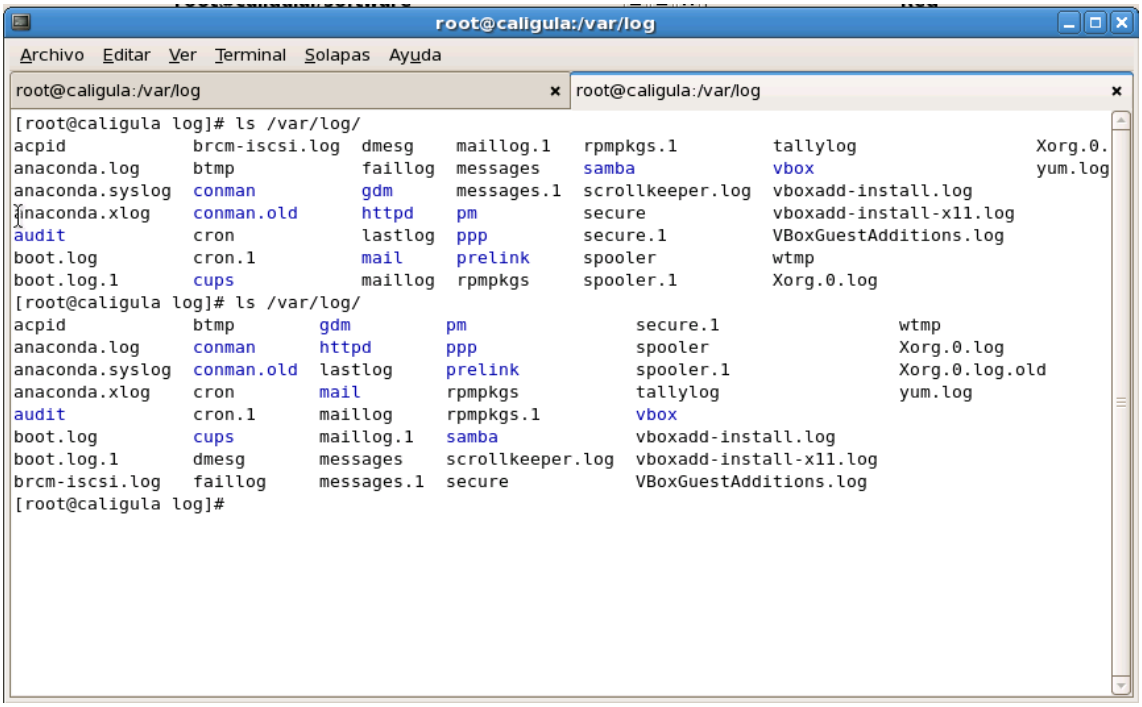
Queda como ejercicio para el lector el añadir el resto de nodos del cluster (storage, nodo1, nodo2…) a Nagios. Pistas:

- Los nuevos hosts y servicios se añaden modificando los archivos de configuración de nagios.
- Revisar archivo: "/opt/nagios/etc/objects/localhost.cfg
- Será necesario crear un nuevo archivo en "/opt/nagios/etc/objects/* para cada nuevo host a añadir.
- Será necesario modificar el archivo "/opt/nagios/etc/nagio.cfg" para cada nuevo host a añadir.
- Cada vez que se modifica un archivo de configuración, Nagios debe comprobar la sintaxis del mismo y recargarla. Para ello se puede reiniciar el servicio "nagios" o al menos recargarlo:
    - */etc/init.d/nagios reload*
    - */etc/init.d/nagios restart*

Sistema de logs centralizado.

Unos de los mayores amigos de un administrador de sistemas Unix son los archivos log.

Desde los inicios de Unix, se ha mantenido la buena costumbre de hacer que tanto el propio sistema operativo como los programas y aplicaciones que corren sobre él generen de forma continua una bitácora con el estado en que se encuentran, errores, etc. Estos archivos se llaman comúnmente "logs" y suelen consistir en archivos de texto, al que se añade una nueva línea por cada evento y que residen habitualmente en el directorio "/var/log" del sistema.



Tanto el sistema operativo como las aplicaciones deben disponer de ficheros de log y es una buena práctica generarlos al crear nuevos programa.

Así, se pueden encontrar, por ejemplo:

/var/log/dmesg: Información directa del kernel

/var/log/messages: Información del kernel y aplicaciones de sistema

Etc…

Pero como se ha dicho, también hay archivos de log específicos de aplicaciones y generados por elloas:

/var/log/maillog: Información del servicio de correo.

Los logs se utilizan comúnmente para detectar y solucionar problemas en el sistema. Si el sistema operativo y las aplicaciones están correctamente escritas, escribirán en sus logs lo que van haciendo y los errores que van encontrando.

Cuando se dispone de múltiples máquina que mantener y administrar, en ocasiones resulta una tarea pesada ir una a una comprobando sus logs. Por eso, es habitual instalar un servidor de logs.

Los logs del sistema pueden ser manejados (y de hecho lo son en todos los sistemas Linux modernos) por un servicio que se encarga de enviarlos a los archivos correspondientes, rotarlos, etc. Estos servicios tienen una funcionalidad interesante, que es la de poder enviar los logs a través de la red a una máquina remota que se encargue de consolidarlos.

En este caso, se explicará como instalar un servidor central de logs para el cluster, en el nodo manager y como configurar los nodos de cálculo para que envíen los logs de sistema más

importantes al mismo. De esta forma es posible administrar y revisar los logs de todos los nodos del cluster desde una misma máquina.

La instalación tiene dos pasos:

1. **Instalar el servidor**
2. **Instalar los clientes**

En este caso se usará un demonio de logs llamado “syslog-ng”. El mismo software sirve como cliente y como servidor así que el mismo paquete serviría para todos los roles.

Sin embargo, por defecto CentOS utilizar el software “syslogd”, con funcionalidades similares pero un diseño más antiguo. En la solución se utilizará “syslog-ng” para el manager como servidor de logs y “syslogd” como cliente en todos los nodos de cálculo.

Instalación del servidor “syslog-ng” en el manager:

Detener el servicio “syslogd” actual y desactivar su arranque:

Service syslog stop

Chkconfig syslog off

Descargar los paquetes de “syslog-ng” , compilarlo e instalarlo:

Descargar:

syslog-ng_3.2.4.tar.gz

eventlog_0.2.12.tar.gz

Nota: “eventlog” es una dependencia de “syslog-ng”

Compilar e instalar bajo «/opt/syslog-ng»

mkdir syslog-ng

mv eventlog_0.2.12.tar.gz syslog-ng

mv syslog-ng_3.2.4.tar.gz syslog-ng

cd syslog-ng/

tar xvfz eventlog_0.2.12.tar.gz

tar xvfz syslog-ng_3.2.4.tar.gz

mkdir -p /opt/syslog-ng/eventlog

cd eventlog

cd eventlog-0.2.12/

./configure --prefix=/opt/syslog-ng/eventlog

make

make install

export PKG_CONFIG_PATH=/opt/syslog-ng/eventlog/lib/pkgconfig/

cd ..

cd syslog-ng

cd syslog-ng-3.2.4/

./configure --prefix=/opt/syslog-ng/

make

make install

ls /opt/syslog-ng

cd /opt/syslog-ng/

mkdir var

Editar el archivo de configuración «/opt/syslog-ng/etc/syslog-ng.conf»

```
@version: 3.2
@include "scl.conf"
options {
    chain_hostnames(0);
    time_reopen(10);
    time_reap(360);
    log_fifo_size(2048);
    create_dirs(yes);
    perm(0640);
    dir_group(adm);
    dir_perm(0750);
    use_dns(yes);
```

```
        stats_freq(0);
        bad_hostname("^gconfd$");
};

source s_network {
        udp();
};

destination d_local {
            #Lugar donde se almacenarán los logs que se reciban.
        file("/var/log/HOSTS/$HOST/$YEAR/$MONTH/$DATE_$FACILITY");
};

log {
        # uncomment this line to open port 514 to receive messages
        source(s_network);
        destination(d_local);
};
```

Arrancar el servicio y activar el arranque automático

```
#service syslog-ng start
#chkconfig syslog-ng on
```

Ahora, el nodo manager almacenará en "/var/log/HOSTS" todos los logs que le lleguen vía IP/UDP.

Configuración de los nodos

Solamente se van a enviar al servidor los logs que normalmente irían al logs archivos locales:

/var/log/secure: Entradas de usuarios al sistema y modificaciones de cuentas.

/var/log/messages: Información genérica del sistema

Para ello solamente es necesario modificar el archivo de configuración de “syslogd”, “/etc/syslog.conf”. Se deben modificar las entradas que se desean enviar al servidor usando la sintaxis:

Log @IPSERVIDOR

Vg.-

Archivo original. Los logs se quedan en local

```
*.info;mail.none;authpriv.none;cron.none        /var/log/messages
authpriv.*                                      /var/log/secure
mail.*                                          -/var/log/maillog

cron.*                                          /var/log/cron
*.emerg                                         *
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

Archivo modificado. Algunos logs se envían al servidor de logs en 10.9.9.11

```
*.info;mail.none;authpriv.none;cron.none        @10.9.9.21
authpriv.*                                      @10.9.9.21
mail.*                                          -/var/log/maillog

cron.*                                          /var/log/cron
*.emerg                                         *
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

Una vez modificado el archivo, es necesario reiniciar el servicio “syslogd” o al menos recargar su configuración:

 # service syslogd restart

ó

 # service syslogd reload

SISTEMA DE GESTIÓN DE COLAS:

[In Future Release]

CARACTERIZACIÓN DEL CLUSTER.

Una vez el cluster ha sido construido y es plenamente funcional, llega la hora de comprobar sus capacidades de cálculo.

Es probable que la funcionalidad se a correcta, sea posible ejecutar todo tipo de programas y obtener resultados correctos, pero quizá la velocidad a la que se obtienen resulte demasiado pobre.

La caracterización consiste en una serie de pruebas de carga, estrés, latencia, etc, que ayuden a definir el verdadero potencial de la máquina y ayudar a encontrar cuellos de botella o posibles problemas de hardware, software o configuración.

En un cluster dedicado específicamente al cálculo, parece obvio que el test más interesante es el de medida de rendimiento, pero se va a explicar como realizar algunos más. Serán los siguientes:

Test de rendimiento en cálculo con LINPACK. Dará una idea de la potencia pico de la máquina en FLOPs.

www.netlib.org/**linpack/**

www.netlib.org/**lapack/**

*es.wikipedia.org/wiki/**Linpack***

Linpack o su hermano pequeño Lapack son en la actualidad los dos test más extendidos y aceptados para caracterizar máquinas de cálculo. Son los únicos test aceptados por el Top500 para realizar el ranking mundial de supercomputadores.

La idea de su funcionamiento (Linpack) consiste en contar el tiempo que tarda la máquina en hacer un cálculo largo cuyo número de operaciones puede ser exactamente conocido de antemano. Este es el caso de la resolución de enormes sistemas de N ecuaciones con N incógnitas.

La división entre en tiempo en realizar el cálculo y el número de operaciones necesarias dará como resultado la potencia de la máquina en FLOPs.

El test de linpack es un test sintético, por lo que sus resultados no son aplicables al mundo real, pero su uso intensivo de la unidad de punto flotante de la CPU, la memoria RAM de la máquina y los casi nulos accesos a disco hacen que sea muy fiable para medir el rendimiento pico de un cluster.

Linpack está originalmente escrito como rutinas de C y Fortran, aunque se pueden encontrar implementaciones en muchos otros lenguajes, incluido JAVA; aunque dada la naturaleza del tests, muy cercano al hardware, se recomienda utilizar implementaciones en lenguajes del nivel más bajo posible.

La ejecución de Linpack no es trivial. El programa recibe varios parámetros, entre los que se incluyen el tamaño de la matriz de coeficientes del sistema de ecuaciones.

El uso de parámetros adecuados para la arquitectura sobre la que se va a ejecutar tendrá incidencia directa en los resultados. La destreza está en ajustar al máximo estos parámetros para obtener los mejores resultados posibles sobre la máquina.

Se recomienda consultar los siguientes enlaces:

http://serverfault.com/questions/150243/how-to-get-the-best-linpack-result-and-conquer-the-top500

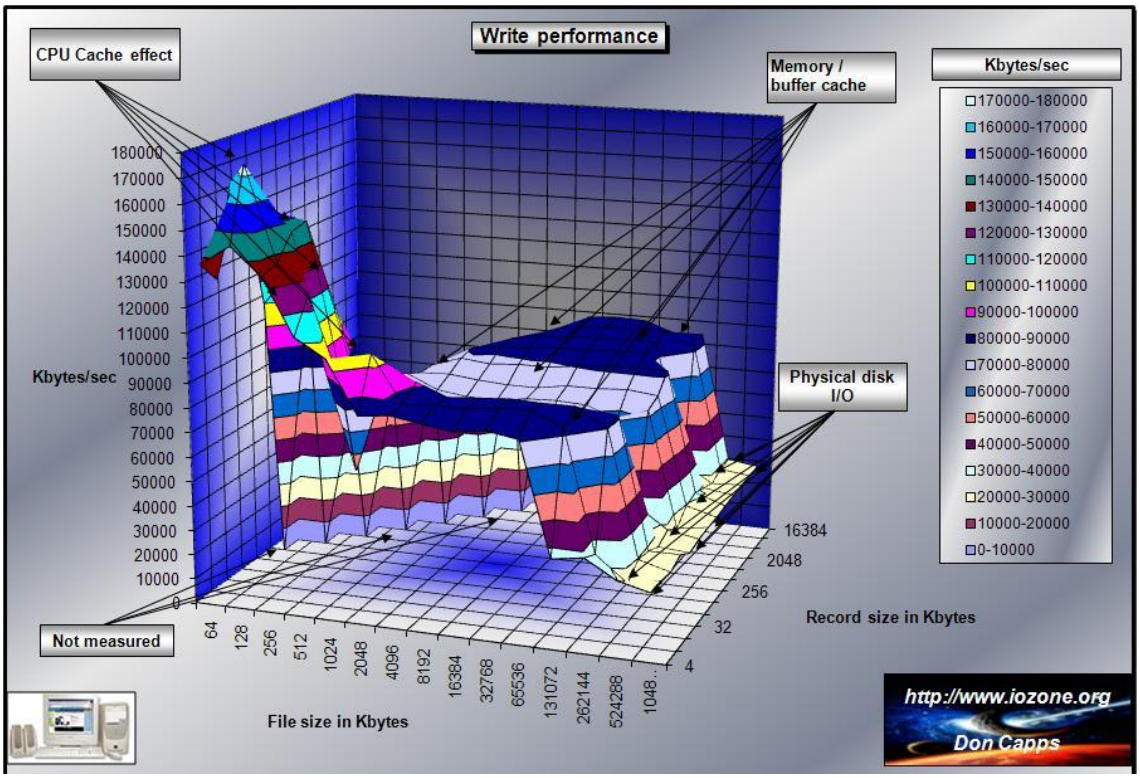
http://www.netlib.org/benchmark/hpl/tuning.html

http://sourceforge.net/projects/hpl-calculator/

Para obtener los mejores resultados se recomienda, ante todo, unas matrices de entrada que saturen todo lo posible la memoria RAM de los nodos pero sin llegar a *swapear*. Entre los enlaces anteriores se encuentra un pequeño programa que ayuda a buscar los valores óptimos para cada cluster.

Test de velocidad de almacenamiento a disco local y NFS con IOZONE

*www.**iozone**.org/*



El acceso a disco es uno de los mayores cuellos de botella en todos los ordenadores actuales. Su velocidad, casi un orden de magnitud más lenta que los buses y las memorias RAM hacen que el acceso a disco termine ralentizando todos los procesos, ya que se producen gran cantidad de estados de espera.

En un cluster, ocurre exactamente lo mismo. Es, de hecho, el acceso a disco el eslabón más débil de la cadena. Optimizar su rendimiento servirá para poder mejorar enormemente el rendimiento general del sistema.

Además, en un sistema de almacenamiento compartido, no solo influye la propia velocidad de giro y transferencia de los discos, sino también la latencia y velocidad de la red que transporta los datos.

Para caracterizar esto se puede utilizar un test, también sintético, llamado *iozone*. Realiza multitud de lecturas y escrituras, con diferentes tamaños de bloque y archivo para determinar tanto el rendimiento general del almacenamiento como sus puntos fuertes y débiles.

En este caso *iozone* debe ejecutarse al menos dos veces. Una que compruebe el acceso a los datos de disco local de los nodos y otra que compruebe el acceso al almacenamiento compartido NFS.

Test de latencia de la red Ethernet.

*sourceforge.net/projects/**iperf/***

Finalmente es fundamental calcular el ancho de banda y la latencia de las redes que comunican los diferentes nodos.

Un técnico de la FCSCL afirma: “Si desapareciera la red de baja latencia en Caléndula, ya no tendríamos un superordenador, sino una máquina muy pesada que consume mucho y hace mucho ruido, pero su rendimiento sería irrisorio”.

Caléndula, el supercomputador de la FCSCL dispone de una red de baja latencia, llamada Infiniband, entre los nodos. Es una tecnología especial para HPC, muy cara, pero con unos resultados espectaculares.

La idea detrás del uso de redes de baja latencia es intentar minimizar la cantidad de estados de espera que se producen cuando un programa paralelo debe esperar a que otro nodo le envíe cierta información que necesita. Si la red que los comunica introduce una gran latencia, esta se añade al estado de espera, arruinando el proceso completo ya que los estados de espera se van acumulando.

Por esto es fundamental conocer la latencia de la red más aún que su ancho de banda, ya que normalmente los bloques de datos intercambiar son muy pequeños (del orden de KBs), pero su trasiego es continuo.

El test de latencia debe estar adecuado a la tecnología de la red que va a testear. En este caso como se utilizan comunicaciones IP sobre Ethernet, se propone una aplicación diseñada para probar comunicaciones de este tipo.

Este test no sería válido en Caléndula, ya que las redes Infiniband no utilizan IP, sino un protocolo llamado RDMA que dispone de tests específicos.

Los test de latencia suelen incluir pruebas del siguiente tipo:

Ping-ping.- Un nodo envía un paquete a otro nodo aleatorio del cluster con un flag especial que lo identifica y otro que indica el número de saltos que ha tenido. El que lo recibe lo reenvía a otro nodo aleatorio. El test se repite hasta que el nodo original lo recibe. En función del número de saltos y el tiempo que tardó en recibirlo puede calcular las latencias.

Los test ping-ping también se realizan en anillo, de forma que el camino del test está predeterminado para que todos los nodos del cluster (o una parte específica del mismo) pasen por la cadena de salto.

Ping-pong.- Es similar al anterior, pero en este caso el nodo que recibe un paquete siempre se lo devuelve al nodo de origen. El nodo que inicia el test puede comprobar, de esta forma, los tiempos que tarda el paquete desde sí mismo a todos los nodos.

La ejecución de test de caracterización deben incluir al menos pruebas de estos tres tipos. Queda a responsabilidad del lector el profundizar en ellas y ejecutarlas sobre sus entornos de pruebas.

OPTIMIZACIONES Y PRUEBAS

Una vez el cluster funciona, siempre va a haber capacidad de mejora en el rendimiento del mismo. Normalmente el proceso de optimización lleva bastante más tiempo que el de instalación y puesta en marcha, pero los beneficios obtenidos en el mismo con suficientes para que valga la pena.

Aunque hay cientos de parámetros que tienen relación directa con el rendimiento de un cluster de cálculo, la mayoría se pueden agrupar en:

- Relacionadas con el hardware del cluster.
- Relacionadas con el software del cluster.
- Relacionadas con el diseño del cluster.

Y todas estarán más o menos relacionadas con:

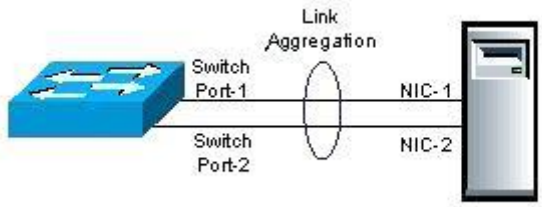
- Latencias de la red de comunicaciones.
- Velocidades de los componentes internos de los nodos.
- Cantidad de memoria RAM de los nodos.
- Calidad del código paralelo del software que se está ejecutando.
- Calidad de las bibliotecas y utilidades MPI que se están utilizando.
- Velocidad de acceso al almacenamiento.
- Sobrecargas en la red
- Programación de los sistemas operativos

Como práctica enormemente educativa, se recomienda realizar diferentes cambios en la arquitectura y configuración del cluster para observar las pérdidas y ganancias de rendiminto.

Se proponen las siguientes prueba:

Uso de *bonding* o agregación en los enlaces de red:

Si se dispone de múltiples interfaces Ethernet de iguales capacidades, es posible configurar los nodos para que agreguen la capacidad de varios de ellos en uno solo.



http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding

La agregación (bonding en Linux), implica duplicar el cableado y el número de interfaces de red. Será interesante descubrir que el bonding sobre Ethernet no mejora significativamente la latencia de la red aunque sí puede llegar a duplicar el ancho de banda. Ya que la latencia es en cálculo paralelo más importante que el ancho de banda, es bastante probable que la mejora no general de rendimiento no sea visible.

Uso de Jumbo Frames

Se llaman "Jumbo Frames" a las tramas Ethernet que contienen más de los 1500 Bytes que suelen tener por defecto en todos los sistemas. Es el llamado parámetro MTU.

*en.wikipedia.org/wiki/**Jumbo_frame***

Si se confía en la red Ethernet subyacente y todos los componentes de la misma lo permiten, es interesante ampliar el MTU hasta los 9000. Esto reduce el *overead* de las tramas, que contienen más carga de datos efectiva.

http://www.cyberciti.biz/faq/rhel-centos-debian-ubuntu-jumbo-frames-configuration/

Esto sí puede mejorar significativamente el rendimiento global del cluster, aunque es necesario que todos los componentes físicos (switches, NICs, etc) como lógicos (OS, Drivers…) puedan usar JF.

Uso de otras tecnologías de red

En el mundo académico es muy popular el uso de Ethernet para construir clusters debido sobre todo a su bajo precio. Sin embargo no es la tecnología más adecuada para este objetivo debido a la propia naturaleza del mismo.

Ethernet es CSMA/CD, lo que quiere decir que, entre otras cosas, todos los nodos que comparten la red están continuamente compitiendo por ella. Las colisiones están a la orden del día y estas son las principales responsables del bajo rendimiento de Ethernet en clusters grandes o en programas paralelos con mucho trasiego de mensajes entre nodos.

Existen otras tecnologías, pensadas específicamente para solventar estos problemas. Las más extendidas son:

Infiniband: *http://en.wikipedia.org/wiki/InfiniBand*

Myrinet: en.wikipedia.org/wiki/Myrinet

Quadrics*: http://en.wikipedia.org/wiki/Quadrics*

A continuación se aconseja la lectura de un artículo que compara el rendimiento entre algunas de ellas:

http://nowlab.cse.ohio-state.edu/publications/conf-papers/2003/liuj-sc03.pdf

Uso de diferentes implementaciones de MPI

Hay multitud de paquetes MPI, tanto propietarios como de código abierto.

Su calidad y estabilidad influyen enormemente en el rendimiento general del cluster, ya que forman diferentes capas de software que están en continua ejecución.

Se recomienda probar otras implementaciones para comparar resultados.

A continuación se incluyen algunos enlaces con pruebas de benchmark sobre diferentes paquetes MPI:

http://www.usenix.org/publications/library/proceedings/als00/2000papers/papers/full_papers/ong/ong_html/

http://hpc.sagepub.com/content/24/4/469.abstract?rss=1

Implementación de un sistema de ficheros paralelo

Cuando NFS no da más de sí, se hace necesario implementar un sistema de ficheros compartido y paralelo que permita balancear la carga de accesos de los nodos sobre diferentes máquinas.

Para esto se han escrito numerosos sistemas de archivos paralelos, entre lo que cabe destacas, dentro del mundo del HPC:

- LUSTRE: *www.lustre.org/*
- *FhGFS:* *www.fhgfs.com/*
- *GPFS:* *www.ibm.com/systems/software/gpfs*
- PVFS: *http://www.pvfs.org/*

En el siguiente enlace puede encontrarse un interesante artículo del fabricante de ordenadores y servidores DELL sobre Parallel-Filesystems:

http://www.dell.com/downloads/global/power/ps2q05-20040179-Saify-OE.pdf

Instalar un sistema de archivos paralelo puede ser tan laborioso como desplegar un cluster MPI, pero cuando el número de nodos del cluster aumenta NFS no consigue satisfacer la demanda de todos ellos. En casos en necesario usar alguno de los arriba mencionados.

Conceptualmente un sistema de archivos paralelo consiste en varias máquinas diferentes que sirven los mismos datos y que permiten balancear la carga de peticiones de lectura y escritura de los nodos entre ellas, permitiendo multiplicar el IO, tasas de trasferencia, etc.

Prácticamente todas las supercomputadoras actuales disponen de un Parallel-fileystem y todas las de la actual lista TOP500.org disponen de ellos.

Caléndula en la FCSCL dispone actualmente de FhGFS.

Optimización del sistema operativo

En el caso concreto del sistema base (CentOS) utilizado en este cluster, es posible modificar diferentes parámetros del kernel y la configuración del sistema para ganar cierto rendimiento.

A continuación se listan algunas de las optimizaciones básicas para nodos de clusters Linux:

- Descargar todo módulo del kernel que no sea totalmente imprescindible. Un comando “lsmod” indicará que módulos están actualmente en uso. Los que no sean necesarios pueden descargarse con el comando “modprobe –r NOMBREMODULO”.
- Para todo servicio del sistema que no sea totalmente imprescindible. Algunas características como los entornos de ventanas (X.org), servicios de impresión (CUPS).

- Elegir correctamente el tamaño de SWAP de los nodos. En ocasiones, será recomendable incluso no utilizar partición de SWAP. Esta memoria de intercambio permite ejecutar programas más grandes que el tamaño de la RAM, pero su velocidad de acceso es tan lenta que en ocasiones es preferible no utilizarla y ejecutar un programa muy grande como dos (o más) de menor tamaño.
- Cambiar el comportamiento del sistema para que solo utilice la memoria SWAP cuando la RAM esté casi completamente llena (95%):

  sysctl vm.swappiness=5

- Establecer límites en el archivo "/etc/security/limits.conf". Este archivo es enormemente complicado de configurar correctamente. Se utiliza para poner los límites a los procesos y usuarios, permitiendo indicar tamaños máximos de memoria usada, tiempos máximos de procesador, número máximo de descriptores abiertos y un largo etc. Multitud de programas MPI requerirán modificaciones en este archivo y será uno de los primero lugares a *tocar* cuando se producen desbordamientos de memoria y problemas similares en ejecuciones MPI.
  http://www.linuxhowtos.org/Tips%20and%20Tricks/ulimit.htm