



CarmenJara

CONTENIDO:

¿PORQUÉ UN CLUSTER? **3**

¿PERO QUÉ ES UN CLUSTER? **3**

ORÍGENES **4**

TIPOS **4**

ALTA DISPONIBILIDAD **4**

PRÁCTICA LINUX **5**

PRÁCTICA WINDOWS **8**

BALANCEO DE CARGA **20**

PRÁCTICA RED HAT **21**

ALTO RENDIMIENTO **28**

PRÁCTICA BEOWULF LINUX **29**

INFORMACIÓN UTILIZADA **35**

CLUSTERS

JUNIO 2015

Página 3

¿PORQUÉ UN CLUSTER?

Porque hay pocas cosas que suenen más exóticas que un supercomputador.

La idea de reutilización de elementos cotidianos para la creación de un ente computacional "superior" ha llenado nuestra imaginación colectiva, azuzada por numerosa literatura al respecto.

Poder acceder a tecnología avanzada por medio de la "colaboración" conlleva a demás un hermoso mensaje para un mundo generalmente muy frío de máquinas y circuitos.

Y también, porqué no reconocerlo, porque quería trastear con hardware y porque me lo ofreció mi profesor de instalación de sistemas, que es el responsable de que esté más cómoda ante una Shell que ante un menú.

Porque al fin y al cabo reconozco que me he vuelto tras estos años un poco friky.

¿PERO QUÉ ES UN CLUSTER?

Clúster, del inglés cluster, "grupo" o "raíz", se aplica a los conjuntos o conglomerados de ordenadores que se comportan como si fuesen una única computadora.

Son empleados para mejorar el rendimiento y/o la disponibilidad por encima de la que provee un solo computador.

Son además, más económicos que computadores individuales de rapidez y disponibilidad comparables.

Para que un clúster funcione como tal, no basta solo con conectar entre sí los ordenadores, sino que es necesario proveer un sistema o software que se encargue de la interrelación

entre las máquinas y los procesos que corren, así como interactuar con el usuario .



LOS TIPOS

Aunque los catalogaremos y dividiremos en tres tipos, no podemos obviar que las combinaciones entre ellos, unidas a las distintas redes y medios compartidos, permiten crear infinidad de variantes.

Alta disponibilidad, conjunto de dos o más máquinas que comparten servicios y que se monitorizan constantemente entre sí con objeto de sustituirse ante caídas.

Balanceo de carga, conjunto de dos o más máquinas que comparten el peso del trabajo a realizar entre sí. Cada petición es enviada a uno u otro en función de la carga de cada equipo.

Alto rendimiento, diseñado para dar altas prestaciones en cuanto a capacidad de cálculo. Para operaciones paralelizables, que se distribuyen entre los componentes.

ORÍGENES

Gene Amdahl de IBM publicó en 1967 las bases del procesamiento paralelo: la Ley de Amdahl, que describe el aceleramiento que se puede esperar paralelizando una serie de tareas.

El primer producto comercial tipo clúster fue ARChet en 1977, pero sin éxito hasta que en 1984 VAXcluster produce el sistema operativo VAX/VMS.

ALTA DISPONIBILIDAD

La principal característica de un clúster de alta disponibilidad es mantener una serie de servicios compartidos y que cada uno de los nodos que forman el clúster sepa en todo momento el estado del otro. Este clúster debe tener un sistema de comunicación, el software del clúster, entre hosts para su correcta monitorización, así como un método para los servicios de un host concreto, que permite que se desplacen entre diversos nodos de manera transparente para la aplicación y los usuarios.

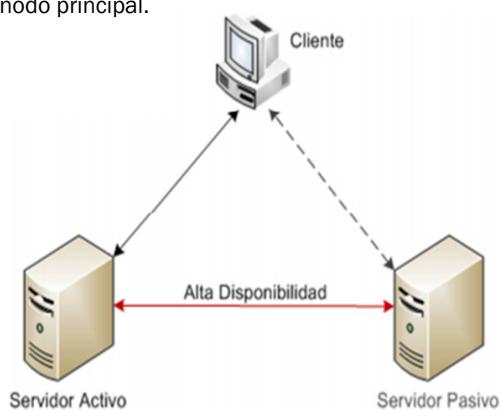
Las configuraciones más comunes en entornos de clústeres de alta disponibilidad son las configuraciones activo/activo y la configuración activo/pasivo.

Configuración activo/activo
En esta configuración todos los nodos del clúster

pueden ejecutar los mismos recursos simultáneamente. Los nodos poseen los mismos recursos y pueden acceder a estos independientemente de los otros nodos del clúster. Si un nodo falla y deja de estar disponible, sus recursos siguen estando accesibles a través de los otros nodos del clúster. La principal ventaja de esta configuración es que los nodos en el clúster son más eficientes ya que pueden trabajar todos a la vez. Pero cuando uno de los nodos deja de estar disponible su carga de trabajo pasa a los nodos restantes, esto produce una degradación en el servicio ofrecido.

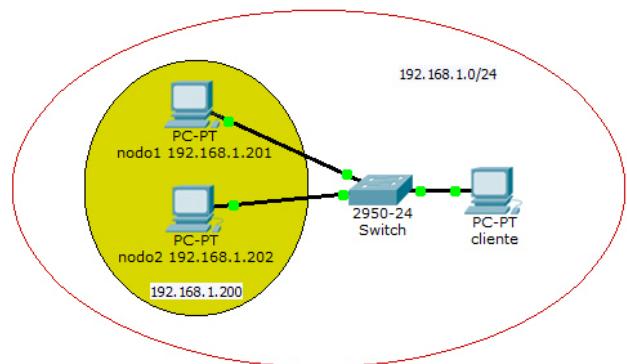
Configuración activo/pasivo
Esta configuración consiste en un nodo principal que posee los recursos del clúster y otros nodos secundarios que son capaces de acceder a estos

recursos, pero no son capaces de tomar el control de estos hasta que el propietario de los recursos ya no está disponible. Esta configuración tiene la ventaja que no hay degradación de servicios, el nivel ofrecido es constante, y que estos servicios solo se reinician cuando el nodo principal deja de estar disponible. La principal desventaja de este sistema con respecto al anterior es que tenemos todos los nodos secundarios ociosos mientras están a la espera del fallo del nodo principal.



PRÁCTICA ALTA DISPONIBILIDAD ACTIVO/PASIVO LINUX

Utilizaremos como nodos del clúster los equipos nodo1 (192.168.1.201) y nodo2 (192.168.1.202) ambos con Ubuntu 10 y como dirección IP virtual (la que usaremos como un recurso en alta disponibilidad) utilizaremos la 192.168.1.200 .



En ambos nodos instalamos el paquete pacemaker para la gestión de recursos del cluster y corosync para la comunicación entre nodos:

nodo1:~# apt-get install pacemaker corosync

nodo2:~# apt-get install pacemaker corosync

Creamos en nodo1 la clave de autenticación de corosync :

nodo1:~# corosync-keygen

```
root@nodo1:/home/carmen
Archivo Editar Ver Terminal Ayuda
root@nodo1:/home/carmen# corosync-keygen
Corosync Cluster Engine Authentication key generator.
Gathering 1024 bits for key from /dev/random.
Press keys on your keyboard to generate entropy.
Press keys on your keyboard to generate entropy (bits = 144).
Press keys on your keyboard to generate entropy (bits = 208).
Press keys on your keyboard to generate entropy (bits = 272).
Press keys on your keyboard to generate entropy (bits = 336).
Press keys on your keyboard to generate entropy (bits = 400).
Press keys on your keyboard to generate entropy (bits = 464).
Press keys on your keyboard to generate entropy (bits = 528).
Press keys on your keyboard to generate entropy (bits = 592).
Press keys on your keyboard to generate entropy (bits = 656).
Press keys on your keyboard to generate entropy (bits = 720).
Press keys on your keyboard to generate entropy (bits = 784).
Press keys on your keyboard to generate entropy (bits = 848).
Press keys on your keyboard to generate entropy (bits = 912).
Press keys on your keyboard to generate entropy (bits = 976).
Writing corosync key to /etc/corosync/authkey.
root@nodo1:/home/carmen#
```

Instalar, si no estuviese, el paquete openssh-server en nodo2 para copiar la clave de corosync en nodo1 al nodo2:

root@nodo1:~# scp /etc/corosync/authkey user2@nodo2:/tmp/authkey

Si es la primera vez que conectamos, nos avisará:

```
The authenticity of host '192.168.1.202 (192.168.1.202)' can't be established.
RSA key fingerprint is f8:9c:43:e6:0f:3a:7f:90:b3:53:3e:3a:14:a6:73:fe.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.202' (RSA) to the list of known hosts.
```

Si ya se han efectuado conexiones previas:

```
root@nodo1:/home/carmen
Archivo Editar Ver Terminal Ayuda
root@nodo1:/home/carmen# scp /etc/corosync/authkey carmen@192.168.1.202:/tmp/authkey
carmen@192.168.1.202's password: authkey
root@nodo1:/home/carmen#
```

Y ya en nodo2, pasamos a su ubicación definitiva:

nodo2:~# cp /tmp/authkey /etc/corosync/authkey

Editamos el fichero de configuración de corosync ([/etc/corosync/corosync.conf](#)) de ambos nodos y añadimos la red que se va a utilizar para controlar el latido entre los nodos.

```
GNU nano 2.2.2      Archivo: /etc/corosync/corosync.conf      Modificado
# Optionally assign a fixed node id (integer)
# nodeid: 1234

# This specifies the mode of redundant ring, which may be none, active,$
rrp_mode: none I

interface {
    # The following values need to be set based on your environment
    ringnumber: 0
    bindnetaddr: 192.168.1.0
    mcastaddr: 226.94.1.1
    mcastport: 5405
```

Línea 42->| bindnetaddr: 192.168.1.0

Además para que corosync se inicie de forma automática al arrancar la máquina, marcamos a “yes” el parámetro START del fichero de configuración del demonio ([/etc/default/corosync](#)) en ambos nodos.

```
root@nodo2: /home/carmen
Archivo Editar Ver Terminal Ayuda
GNU nano 2.2.2      Archivo: /etc/default/corosync

# start corosync at boot [yes|no]
START=yes
```

Reiniciamos corosync en los dos nodos:

nodo1:~# service corosync restart
nodo2:~# service corosync restart

Comprobamos el estado del cluster:

root@nodo1:~# crm_mon

OJO!!

Un error común, especialmente si estamos utilizando virtualización y clonamos las máquinas, es que los equipos tengan igual nombre.
Podemos modificar en [/etc/hosts](#) y [/etc/hostname](#)

Al lanzar el comando de monitorización, crm_mon, veremos en primer lugar lo siguiente:

```
root@nodo1: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May 9 13:39:21 2015
Current DC: NONE
0 Nodes configured, unknown expected votes
0 Resources] configured.
```

OJO!!

En la práctica, hemos definido el recurso como asociado a eth2, por tanto, en ambos nodos, debemos tener eth2 operativa.

Puede ser cualquier tarjeta de red, pero en cualquier caso, en todos los nodos deben tener la misma que definamos en el recurso compartido.

pero al cabo de unos instantes, pasamos al siguiente estado:

```
root@nodo1: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May 9 13:40:00 2015
Stack: openais
Current DC: nodo1 - partition with quorum
Version: 1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
0 Resources configured.
```

Online: [nodo1 nodo2]

Al monitorizar ahora el cluster, veremos que aparece el recurso FAILOVER-ADDR asociado en este momento a nodo1:

```
root@nodo1: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May 9 13:55:43 2015
Stack: openais
Current DC: nodo1 - partition with quorum
Version: 1[0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
1 Resources configured
```

Online: [nodo1 nodo2]

FAILOVER-ADDR (ocf::heartbeat:IPaddr2): Started nodo1

Podemos comprobar que los dos nodos se han detectado, aunque todavía no hay ningún recurso configurado.

El recurso que vamos a configurar en este ejemplo va a ser una dirección IP 192.168.1.200, para ello en primer lugar desactivamos el mecanismo de STONITH (Shoot The Other Node In The Head), que se utiliza para parar un nodo que esté dando problemas y así evitar un comportamiento inadecuado del cluster. Ya que nuestro cluster es de dos:

nodo1:~# crm configure property stonith-enabled=false

```
root@carmen-laptop1:~# crm configure property stonith-enabled=false
INFO: building help index
root@carmen-laptop1:~#
```

Ahora configuramos el recurso de la IP virtual:

```
nodo1:~# crm configure primitive FAILOVER-ADDR
ocf:heartbeat:IPaddr2 params ip="192.168.1.200" nic="eth2"
op monitor interval="10s" meta is-managed="true"
```

Desde un equipo de la red, en nuestro caso usaremos el host Windows, probamos a hacer ping a la 192.168.1.200 y verificamos que la dirección MAC corresponde al nodo1:

host:~\$ ping -c 1 192.168.1.200
Host:~\$ arp -a

```
C:\Users\carmen>ping 192.168.1.200
Haciendo ping a 192.168.1.200 con 32 bytes de datos:
Respueta desde 192.168.1.200: bytes=32 tiempo=1ms TTL=64
Respueta desde 192.168.1.200: bytes=32 tiempo=1ms TTL=64
Respueta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64

Estadísticas de ping para 192.168.1.200:
Paquetes: enviados = 3, recibidos = 3, perdidos = 0
<0% perdidos>
Tiempos aproximados de ida y vuelta en milisegundos:
Mínimo = 1ms, Máximo = 2ms, Media = 1ms
Control-C
C:\Users\carmen>arp -a
Interfaz: 192.168.1.37 --- 0x6
          Dirección de Internet           Dirección física      Tipo
192.168.1.1    00-19-15-d5-b9-5c    dinámico
192.168.1.34   84-34-97-0e-05-0c    dinámico
192.168.1.76   c0-3f-d5-ef-7b-60    dinámico
192.168.1.200  08-00-27-c9-d0-37    dinámico
192.168.1.201  08-00-27-c9-d0-37    dinámico
192.168.1.255 ff-ff-ff-ff-ff-ff    estático
```

Ahora probamos el funcionamiento del cluster apagando nodo1 y monitorizamos el cluster desde nodo2.

Vemos que detecta la baja de nodo 1. sin embargo nodo2 no pasa a ofrecer el recurso directamente.

```
root@nodo2: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May  9 15:07:03 2015
Stack: openais
Current DC: nodo2 - partition WITHOUT quorum
Version: 1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
1 Resources configured.
=====
Online: [ nodo2 ]
OFFLINE: [ nodo1 ]
```

Esto ocurre porque no hay cuórum en el cluster. El cuórum (quorum) es una propiedad que utiliza pacemaker para tomar las decisiones apropiadas mediante consultas consensuadas a todos los nodos, pero no tiene razón de ser en un cluster de sólo dos nodos, ya que sólo habrá quorum cuando los dos nodos estén operativos. Obligaremos por tanto, a ignorar las decisiones basadas en cuórum:

nodo2:~# crm configure property no-quorum-policy=ignore

```
root@nodo2: /home/carmen
Archivo Editar Ver Terminal Ayuda
root@nodo2:/home/carmen# crm configure property no-quorum-policy=ignore
INFO: building help index
root@nodo2:/home/carmen#
```

Podemos comprobar ahora como nodo2 pasa a ofrecer el recurso.:

```
root@nodo2: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May  9 15:13:26 2015
Stack: openais
Current DC: nodo2 - partition WITHOUT quorum
Version: 1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
1 Resources configured.
=====
Online: [ nodo2 ]
OFFLINE: [ nodo1 ]
FAILOVER-ADDR (ocf::heartbeat:IPAddr2): Started nodo2
```

Desde el equipo externo podemos comprobar que la dirección IP sigue respondiendo al ping, pero ahora está asociada a la dirección MAC de nodo2.

host:~\$ ping -c 1 192.168.1.200
host:~\$ arp -a

```
C:\Users\carmen>ping 192.168.1.200
Haciendo ping a 192.168.1.200 con 32 bytes de datos:
Respuesta desde 192.168.1.200: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64

Estadísticas de ping para 192.168.1.200:
    Paquetes: enviados = 2, recibidos = 2, perdidos = 0
    <0% perdidos>
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 2ms, Media = 1ms
Control-C
C:\Users\carmen>arp -a
Interfaz: 192.168.1.37 --- 0x6
          Dirección de Internet      Dirección física      Tipo
 192.168.1.1                      00-19-15-d5-b9-5c  dinámico
 192.168.1.34                     84-34-97-0e-05-0c  dinámico
 192.168.1.76                     c8-31-d5-e4-2b-60  dinámico
 192.168.1.200                     08-00-27-35-60-14  dinámico
 192.168.1.202                     08-00-27-35-60-14  dinámico
 192.168.1.203                     08-00-27-35-60-14  dinámico
```

Si por último, volvemos a arrancar nodo1 y comprobamos como el nodo1 pasa a ser el servidor del recurso:

```
root@nodo2: /home/carmen
Archivo Editar Ver Terminal Ayuda
=====
Last updated: Sat May  9 15:19:54 2015
Stack: openais
Current DC: nodo2 - partition with quorum
Version: 1.0.8-042548a451fce8400660f6031f4da6f0223dd5dd
2 Nodes configured, 2 expected votes
1 Resources configured.
=====
Online: [ nodo1 nodo2 ]
FAILOVER-ADDR (ocf::heartbeat:IPAddr2): Started nodo1
```

Sintaxis del comando crm
<http://crmsh.github.io/man/>

Possible mejora a la configuración
Es recomendable que los nodos estén interconectados por interfaces dedicadas en otro segmento de red diferente del que se utiliza para ofrecer los recursos.

PRÁCTICA ALTA DISPONIBILIDAD WINDOWS

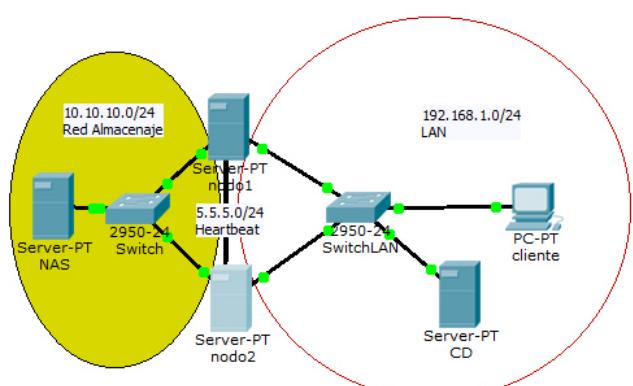
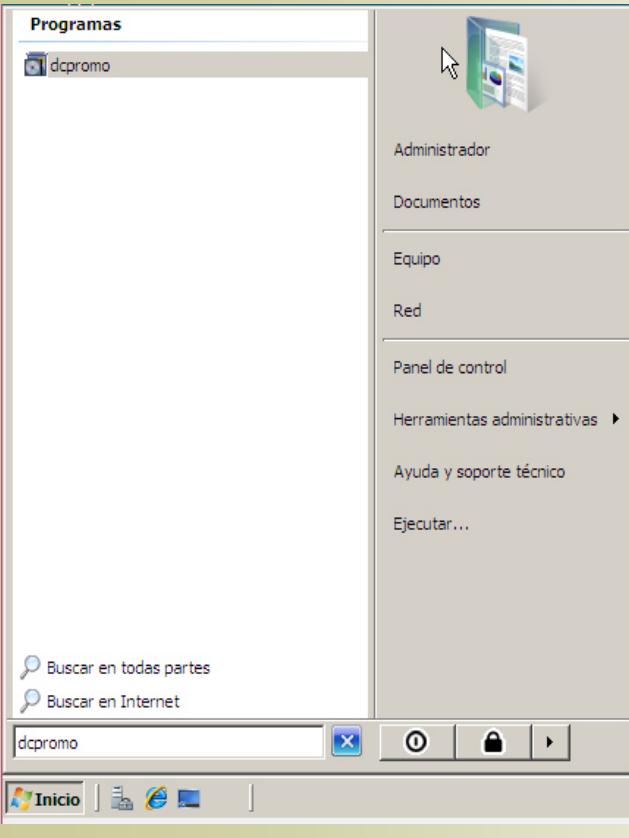
Utilizaremos un Controlador de dominio Windows Server 2008 y dos Windows Server 2008 Enterprise como nodos del cluster de tipo failover clustering.

Los nodos disponen de tres interfaces de red, una la red (192.168.1.0/24) que les conectarán con los equipos de la red, otra la pata de la red de almacenaje (10.10.10.0/24) que conecta a los nodos con el almacenamiento, y finalmente una red HeartBeat (5.5.5.0/24) que servirá para comunicarse entre sí a los nodos del clúster y tener notificación cuando se caiga uno de ellos.

El cluster ofrecerá servicio de ficheros. Para ello se configurarán unos discos como recurso compartido en un equipo con OpenNAS.

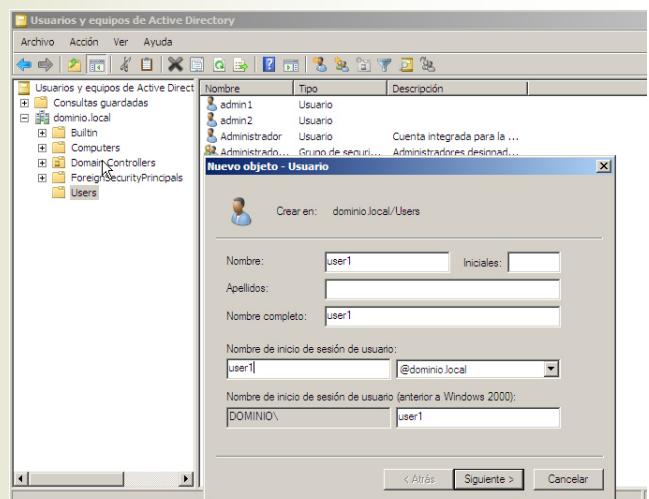
El mantenimiento de estos clústers es costoso, por tanto, procuraremos que ambos servidores tengan las mismas aplicaciones instaladas, de la misma forma y con las mismas versiones, todo esto para evitar problemas futuros. Además es recomendable balancear el clúster cada cierto tiempo para comprobar su correcto funcionamiento.

Creamos dominio: **dominio.local** ejecutando **dcpromo**



Seguiremos los pasos del asistente, eligiendo crear un dominio nuevo en bosque nuevo y aceptando la instalación de DNS.

Creamos a "user1", que nos servirá para las comprobaciones finales del cliente, como **usuario de active directoy**.



Agregamos los nodos al dominio y los preparamos para el cluster.

OJO!!

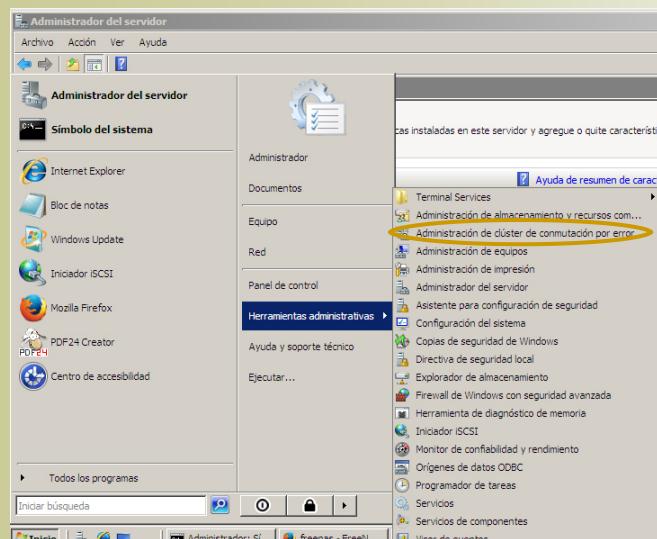
Si se trabaja con virtualización y se han clonado las máquinas se debe ejecutar sysprep.exe en C:\Windows\System32\sysprep en modo auditoría generalizada para que no haya conflicto.

Como hemos comentado en la introducción de la práctica, los nodos deben ser WS2008 Enterprise, ya que la característica de clúster de conmutación por error no está disponible en Windows Web Server 2008 ni en Windows Server 2008 Standard.

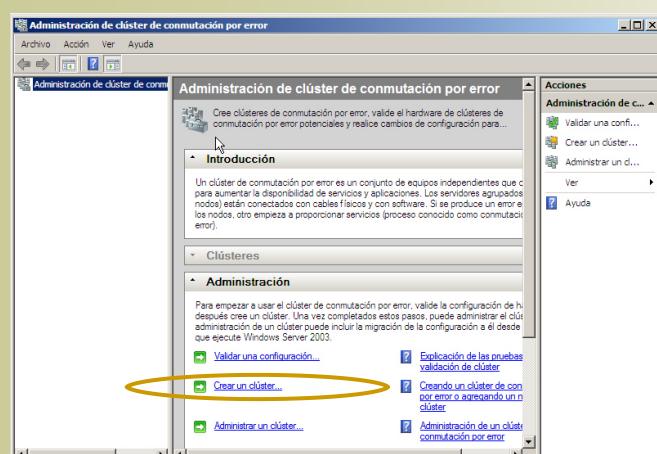
Iniciamos sesión en nodo1 y en nodo 2 como administrador del dominio y desde inicio, Herramientas administrativas, Administrador del servidor, vamos a Agregar características “cluster de conmutación por error”.

Seguiremos el asistente hasta completar la instalación.

Una vez efectuado, desde el nodo1 abrimos la consola que tendremos dentro de las “Herramientas Administrativas” llamada “Administración de clúster de conmutación por error”.

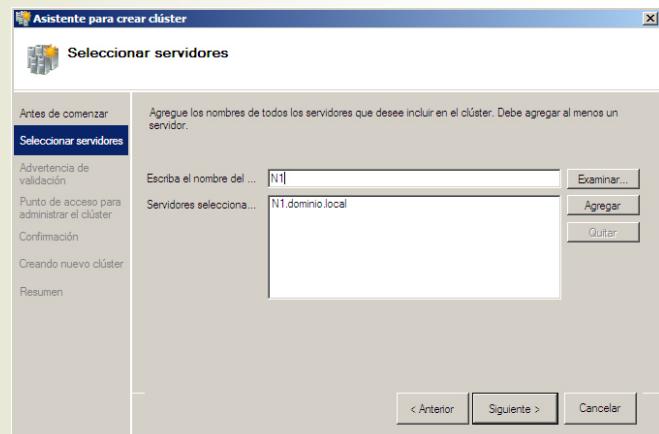


Pulsamos en “Crear un clúster...” para iniciar el asistente.



Ejecutamos siempre TODAS las pruebas.

En el asistente indicamos el nombre de un nodo que queremos que pertenezca a este nodo, en principio, nosotros mismos.



Permitimos tal como ya dijimos, ejecutar pruebas, que es la opción que viene marcada por defecto:



Pinchamos finalizar.

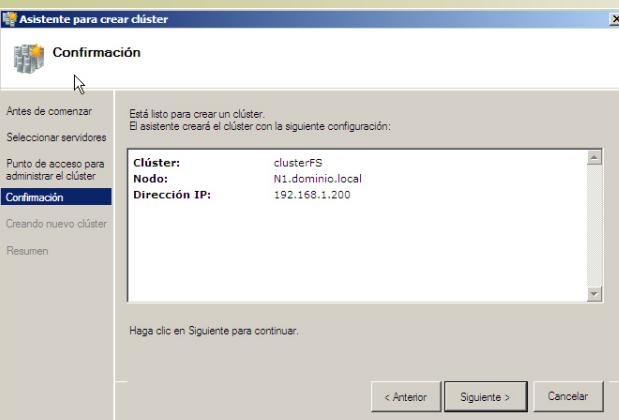
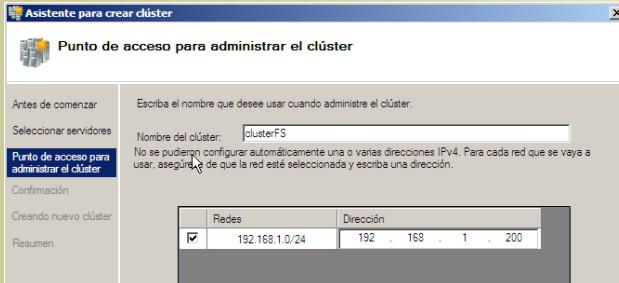
OJO!!

Si nos indica que existe un error caché, quitar y volver a poner la característica de cluster.

Si no se soluciona, modificar el registro como administrador del dominio, con regedit :

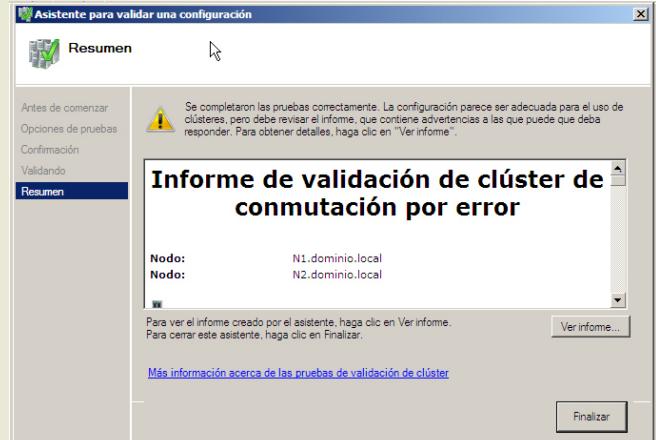
<http://support.microsoft.com/kb/973838/en-us>

Nos pedirá ahora el asistente un nombre al clúster, y una dirección IP de gestión, confirmaremos y tras comprobar el resumen finalizaremos.

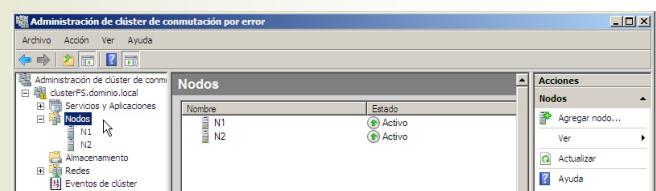


Una vez creado, podemos ir añadiendo los nodos al clúster, para ello pulsamos en "Aregar nodo..."

Mediante el asistente agregamos tantos nodos como queramos que tenga el clúster, indicando el nombre del nodo nuevo.



Una vez que dispongamos de los nodos del cluster, como podemos comprobar:

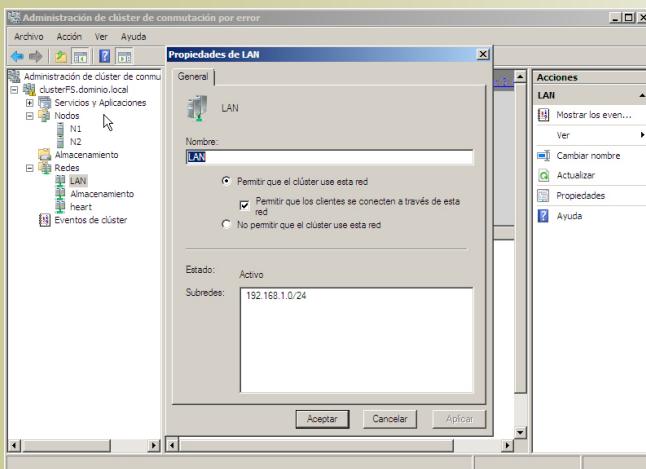


Cambiaremos el nombre de las redes para su mejor identificación, desplegando el árbol Redes de la izquierda en la consola del cluster y pinchando en la derecha "cambiar nombre":

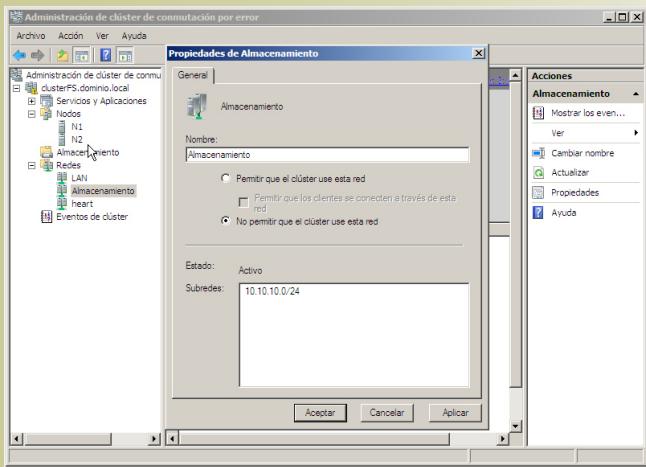


Cuando ya les hemos cambiado el nombre a las redes, indicando la función que tendrán, las configuramos desde sus "Propiedades".

Sobre cada tipo de red, deberemos configurar si daremos acceso al clúster a esta red, si daremos acceso a los clientes a esta red o si no permitiremos que el clúster use la red. En este caso, en la red LAN, permitirémos lógicamente que el clúster acceda a ella para dar servicio a los clientes, marcando "Permitir que el clúster use esta red" y "Permitir que los clientes se conecten a través de esta red".



En la red almacenamiento será diferente, ya que es la red que usará cada nodo para conectarse al almacenamiento, así que el clúster aquí no es necesario, marcamos "No permitir que el clúster use esta red".



Y a la red HeartBeat, la dejamos como está por defecto, dando acceso al clúster, aunque esto es opcional.

Una vez tenemos la red configurada, vamos a configurar el almacenamiento, esto es, añadir los discos que tenemos conectados o bien por fibra o bien por iSCSI a los nodos.

Para ello, usaremos el botón derecho en "Almacenamiento" y elegiremos "Agregar un disco",

Pero necesitamos un disco propio y específico para ello, y además Windows nos exige por fuerza que así sea. No podemos, tal como haremos posteriormente en otras prácticas Linux, duplicar la información en un disco en cada nodo.

Podríamos elevar las funciones de un cuarto Windows server, pero además de encarecer la instalación, WStorage necesita 64 bits.

Probamos por tanto a realizar el disco compartido por medio de FreeNAS. FreeNAS es un sistema operativo basado en FreeBSD que proporciona servicios de almacenamiento en red. NAS son las siglas en inglés de Almacenamiento Conectado en Red (Network Attached Storage).

Este sistema operativo gratuito, open-source y software libre permite convertir un PC personal en un soporte de almacenamiento accesible desde red.

Instalamos FreeNAS 9.2.1.9, que es una versión que permite 32 bits en una máquina virtual BSD.

Tras su instalación, aparecerá un menú donde elegimos la opción 1 para configurar la red:

```

Sun May 10 04:50:50 PDT 2015
FreeBSD/i386 (freenas.local) (ttyu0)

Console setup
-----
1) Configure Network Interfaces
2) Configure Link Aggregation
3) Configure VLAN Interface
4) Configure Default Route
5) Configure Static Routes
6) Configure DNS
7) Reset WebGUI login credentials
8) Reset to factory defaults
9) Shell
10) Reboot
11) Shutdown

You may try the following URLs to access the web user interface:
http://0.0.0.0

Enter an option from 1-11: 1

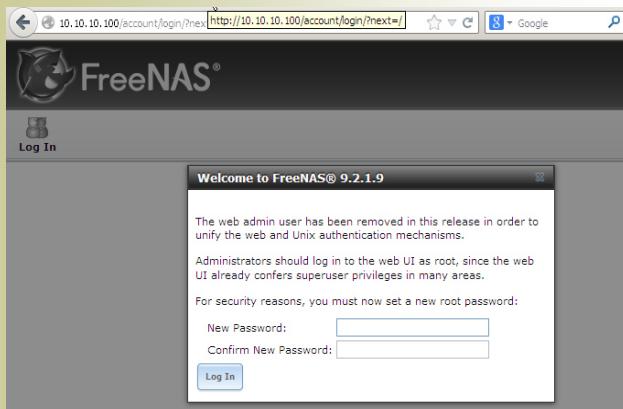
Enter an option from 1-11: 1) pcn0
Select an interface (q to quit): 1
Reset network configuration? (y/n) n
Configure interface for DHCP? (y/n) n
Configure IPv4? (y/n) y
Interface name:pcn0
Several input formats are supported
Example 1 CIDR Notation:
        192.168.1.1/24
Example 2 IP and Netmask seperate:
        IP: 192.168.1.1
        Netmask: 255.255.255.0, /24 or 24
IPv4 Address:May 10 04:59:21 freenas ntpd_initres[1928]: host name not found: 0.freebsd.pool.ntp.org
May 10 04:59:21 freenas ntpd_initres[1928]: host name not found: 1.freebsd.pool.ntp.org
May 10 04:59:21 freenas ntpd_initres[1928]: host name not found: 2.freebsd.pool.ntp.org
10.10.10.100/24
Saving interface configuration: Ok
Configure IPv6? (y/n) n

```

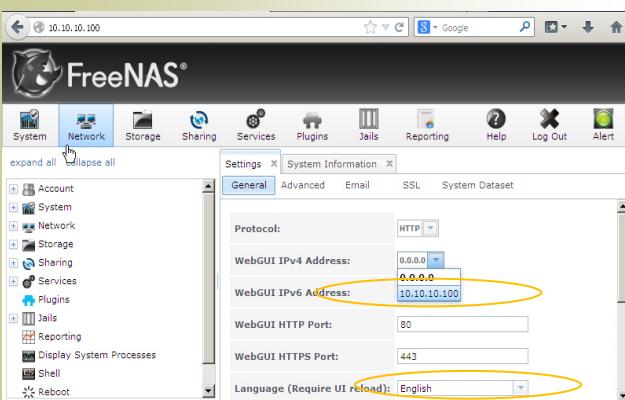
Una vez configurada la IP, podremos acceder a la administración vía web simplemente indicando en la barra de navegación la IP que hemos definido.

Es aconsejable instalar Firefox, ya que otros navegadores pueden darnos problemas con la interfaz.

Como vemos, nada más conectarnos vía web nos pide que definamos una contraseña para root y una vez hecho, nos ofrece una interfaz con menús.



Comenzamos por tanto la configuración de FreeNAS, modificando en la pestaña "Setting" la IP que le hemos asignado, ya que por defecto nos vendrá definida como 0.0.0.0, y también el lenguaje para la interfaz de administración. Salvamos la configuración en el botón que encontramos en la parte inferior.



Una vez efectuado, nos vamos en el menú lateral a servicios/ISCIS donde crearemos un portal e iniciador.

ISCSI

Consideraciones Básicas de iSCSI

El protocolo iSCSI utiliza TCP/IP para sus transferencias de datos. Al contrario que otros protocolos de red diseñados para almacenamiento, solamente requiere una simple y sencilla interfaz Ethernet (o cualquier otra red compatible TCP/IP) para funcionar.

Esto permite una solución de almacenamiento centralizada de bajo coste sin la necesidad de realizar inversiones costosas ni sufrir las habituales incompatibilidades asociadas a las soluciones canal de fibra para redes de área de almacenamiento.

Los críticos de iSCSI argumentan que este protocolo tiene un peor rendimiento que el canal de fibra por la sobrecarga que generan las cabeceras de paquetes. Sin embargo las pruebas que se han realizado muestran un excelente rendimiento de las soluciones iSCSI SANs.

Para el tráfico de red iSCSI, utilizaremos un segmento de red Ethernet exclusivo e independiente de los segmentos de red utilizados para el tráfico de red normal.

Lo primero que tendremos que configurar en nuestro entorno, son los Targets. Para empezar, debemos tener claro, que en un servidor iSCSI, podemos (y debemos) crear múltiples Target. Pero ¿Qué es un Target?

Un Target es un elemento de configuración, a través del cual asignamos los Discos Virtuales (LUNs) a los equipos clientes (iSCSI initiators).

Como regla general, deberemos crear un Target para cada equipo Cliente, pero en los clusters, no será así.

Para utilizar Discos Virtuales (LUNs) compartidos en un Cluster, es necesario crear un Target y asignar los diferentes nodos del Cluster como clientes (iSCSI initiators) del mismo Target.

En consecuencia, y dado lo comentado en el punto anterior, podemos crear un único Target para el Cluster, dónde asignar todos los Discos Virtuales (LUNs) y clientes (iSCSI initiators), o bien podemos crear un Target para cada Disco Virtual (LUN) asignando en cada Target a cada equipo cliente (iSCSI initiators).

Téngase en cuenta, que el escenario de discos compartidos de un Cluster es el único caso (al menos, habitualmente) en que un Target es compartido por múltiples clientes (iSCSI initiators), ya que si no se tratase de los discos compartidos de un Cluster (shared LUNs), existe un riesgo potencial de conflicto y/o corrupción en el acceso a un mismo Disco Virtual (LUN) por múltiples equipos clientes simultáneamente.

ISCSI

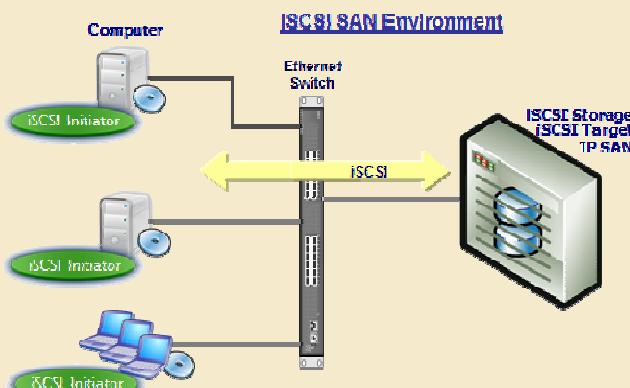
Procedimiento de configuración

En el servidor de almacenamiento, configurar los Targets necesarios. Seguidamente, crear los correspondientes Discos Virtuales (LUNs), asignando cada Disco Virtual (LUN) al Target que le corresponda.

En los equipos clientes (iSCSI initiators) es necesario especificar el servidor de almacenamiento (Portal Target), ya sea indicando su dirección IP o un nombre DNS (y el puerto TCP, por defecto el 3260). Seguidamente, especificamos (opcionalmente, pero recomendable) volver a restaurar la conexión cada vez que el equipo se vuelve a iniciar para así mantener sus unidades (LUNs) persistentemente.

Recordemos, que deberemos crear particiones y formatearlas en los Discos Virtuales (LUNs), ya sea montándolos en el servidor de almacenamiento, o bien formateándolos desde los clientes (iSCSI initiators).

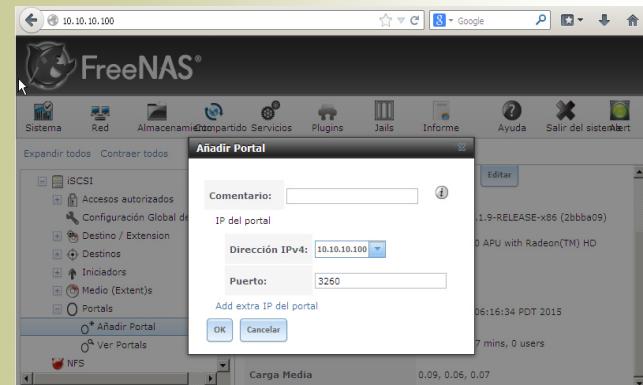
Finalmente, los Discos Virtuales (LUNs) deben quedar asignados a los correspondientes equipos clientes (iSCSI initiators) y montados correctamente (ya sea utilizando una letra de unidad o un punto de montaje).



Algunos profesionales lo castellanizan como "escasi" o "escosi"

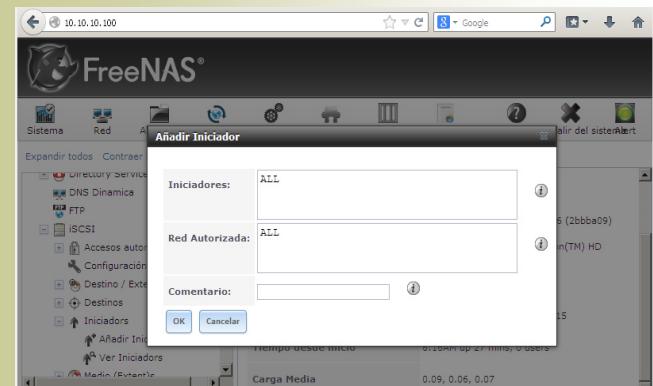
Comenzaremos creando el portal:

Servicios/ISCIS/Portal /añadir portal



Lo siguiente que haremos será añadir un iniciador:

Servicios/ISCIS/Iniciador/añadir iniciador



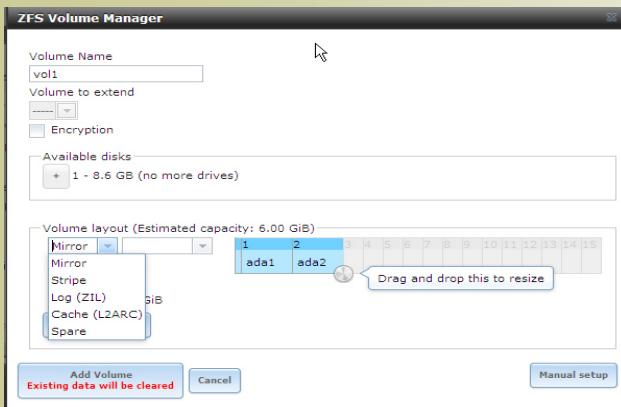
Si aun no los teníamos, es el momento de apagar FreeNAS para **añadirle los discos reales**, o en nuestro caso, VDI desde la aplicación de virtualización, que serán los utilizados para almacenamiento y quorum. Este último espacio es requisito indispensable para el cluster de dos nodos en Windows Server.

Reiniciamos y volvemos a la interfaz web de administración donde podemos ver y comprobar los discos en

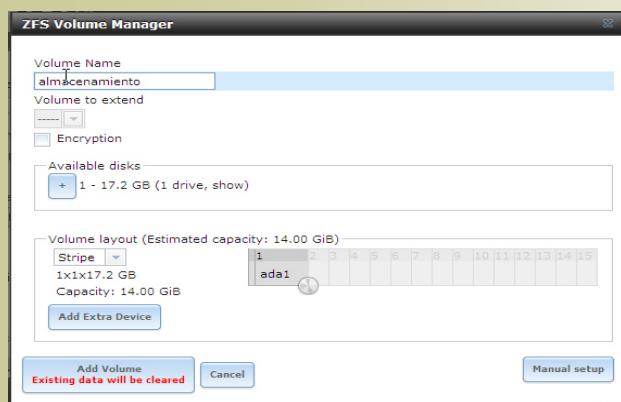
Almacenamiento/volúmenes/ver discos.



En el mismo menú de “volúmenes”, podemos encontrar el gestor ZFS:

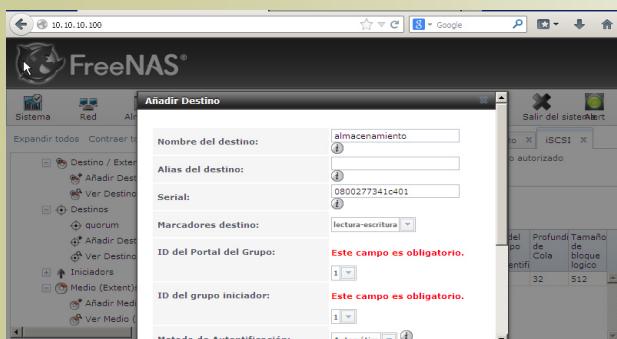


Como vemos, nos da opción de ponerle nombre al volumen que creamos y al darle al signo “+” en “Available Disks”, nos ofrece, en azul, ambos discos y la posibilidad de crear distintos tipos. En nuestro caso escogeremos Stripe (Raid 0), disminuiremos gráficamente a un único disco y pulsaremos el botón Add Volumen, que se encuentra en la parte inferior.



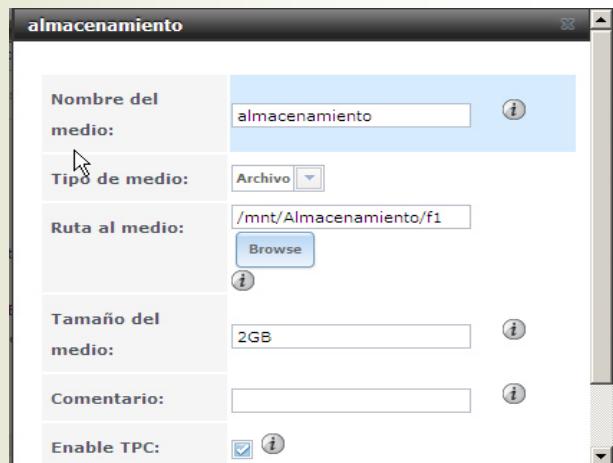
Repetiremos la operación, para crear un segundo volumen.

A continuación, creamos 2 destinos (almacenamiento y quorum) en Servicios/ISCIS/Destino/añadir destino



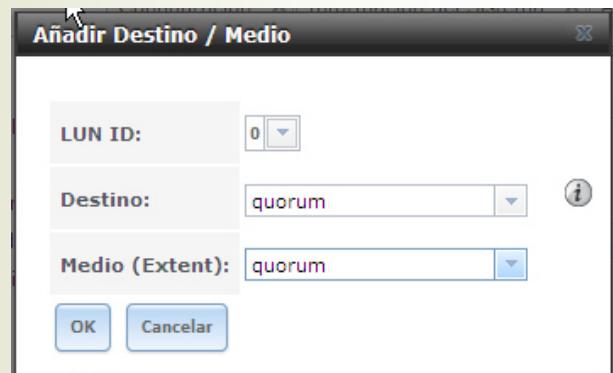
Y posteriormente añadiremos dos medios

Servicios/ISCIS/Medio/añadir medio

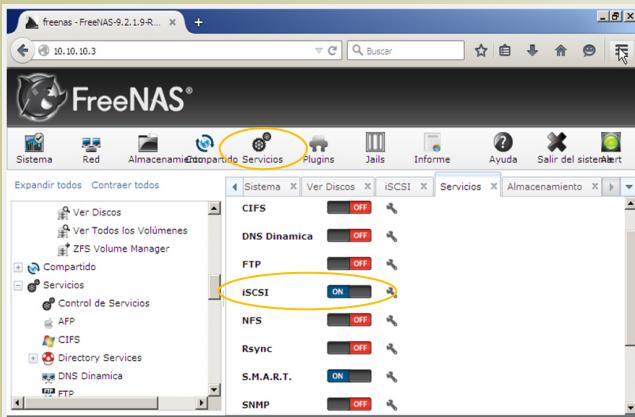


Donde pondremos nombre al medio (usaremos los mismos por comodidad), buscaremos una ruta a uno de los volúmenes en /tmp/, incluiremos un nombre de fichero para crear y le daremos tamaño, en nuestro caso 2GB.

Relacionamos ambos destinos y medios pero indicando distintos LUN ID (al primero 0 y al segundo auto) en Servicios/ISCIS/Destino-extensión/añadir



Por último, iniciamos el servicio iSCSI gráficamente desde la opción Servicios del panel superior:



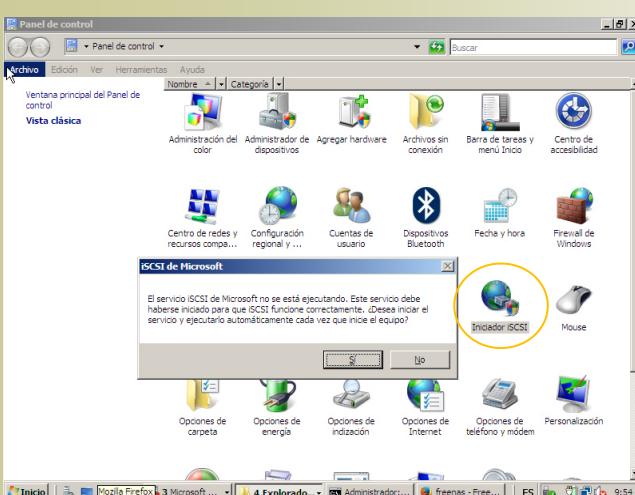
OJO!

Si no se inicia el servicio, es posible que hayamos cambiado alguna proporción en las unidades de medida de información o el ordenamiento de los LUN ID.

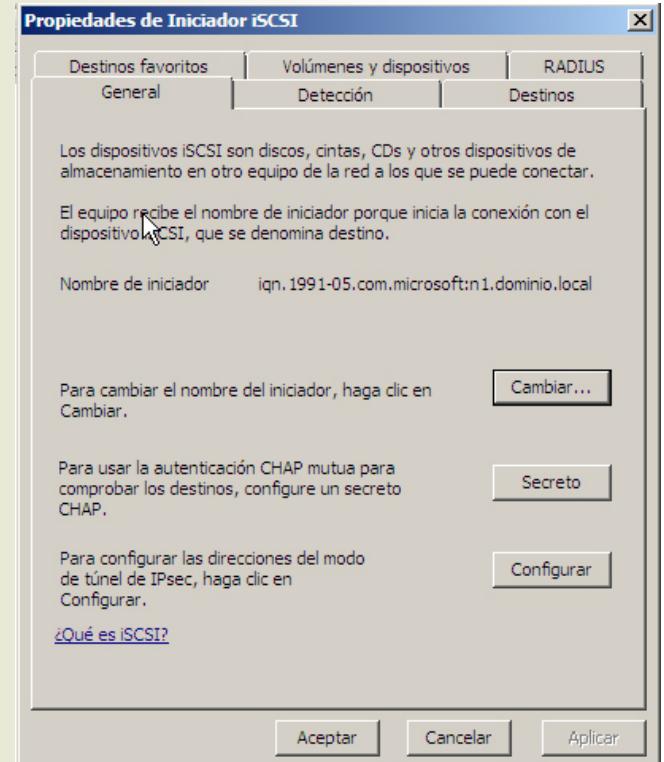
Freenas tiene una estructura de configuración relacional muy compleja que no hemos entrado a desgranar.

Una vez iniciado el servicio iSCSI, volvemos a WS2008.

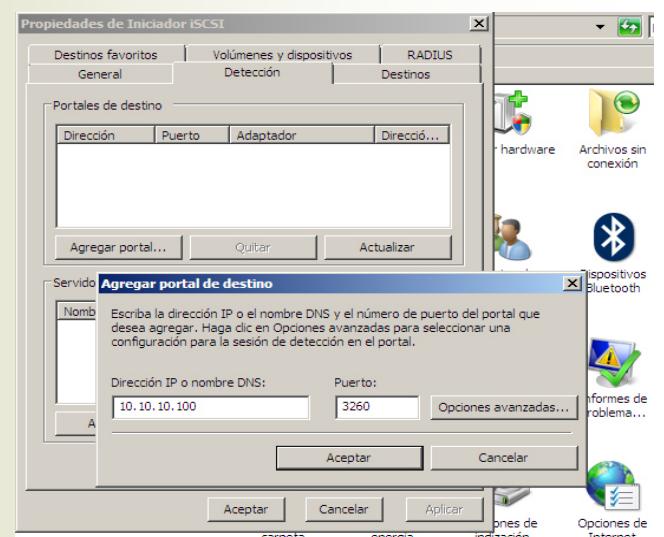
En el panel de control, podemos localizar el “iniciador iSCSI”:



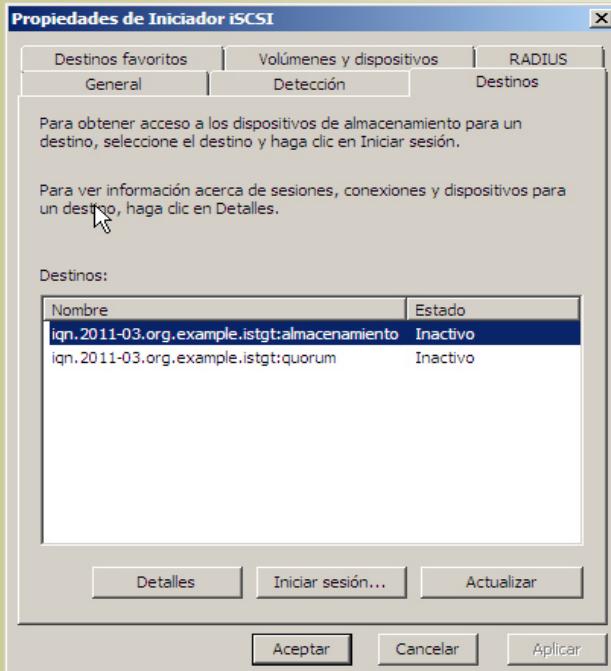
Aceptamos que arranque el servicio al inicio y que se cree una regla que lo permita en el firewall. Finalmente nos aparecerá las propiedades:



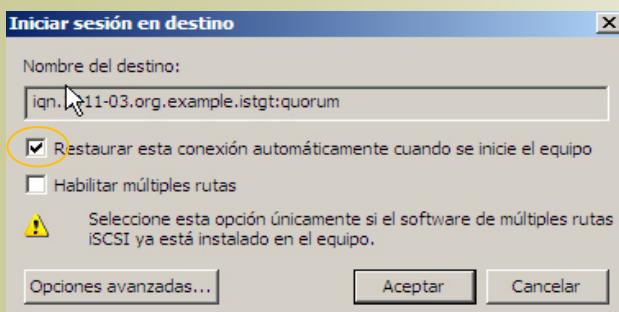
En la pestaña de Detección, pulsamos agregar portal e introducimos la IP de nuestro Freenas, dejando el puerto por defecto que viene indicado, el 3260.



Una vez hecho esto, si pasamos a la pestaña destino, ya podemos ver los nombres de nuestros dos destinos en FreeNAS y podemos iniciar sesión en ambos.



Hemos elegido en el inicio de ambas secciones, iniciar automáticamente:

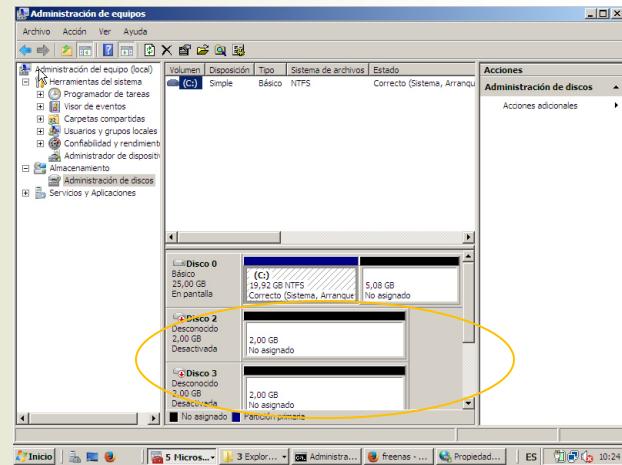


OJO!!

Si desconectamos o apagamos FreeNAS o los nodos antes de la configuración **TOTAL** del cluster, dará problemas de reconexión y en algunos casos será imposible recuperar.

Ahora en el Nodo1, vamos al administrador de discos, donde ya veremos los dos discos del equipo FreeNAS como propios.

Debemos ponerlos en linea, Inicializarlos y crear un nuevo volumen simple en cada uno.



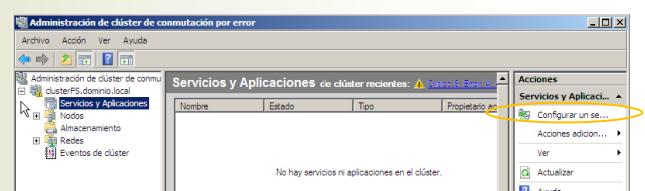
Repetimos la operación en el nodo2. Tanto el iniciador de iSCSI como el administrador de discos, salvo que aquí, los discos aparecen ya formateados como nuevo volumen, por tanto sólo debemos ponerlos en línea:

Ya podemos volver a la consola del cluster en alguno de los dos Nodos. Y desplegando el árbol, en almacenamiento, vamos a seleccionar agregar un disco.



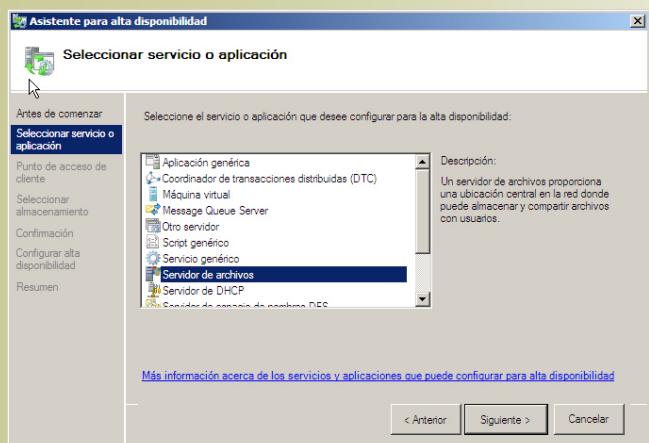
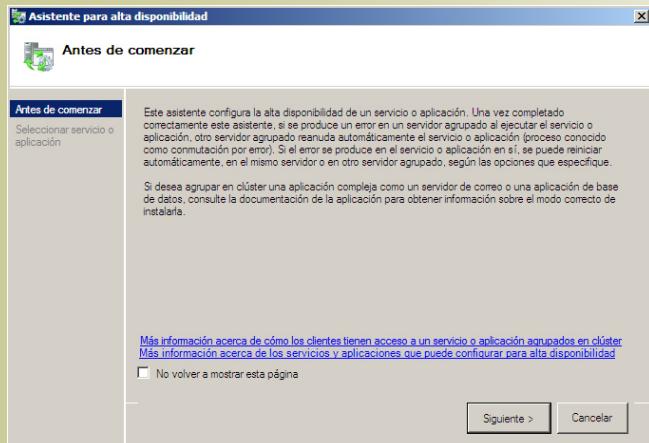
Nos aparecerán los dos discos señalados. Aceptamos y finalizamos.

Ya estamos en condiciones de poder agregar el servicio. Para ello, lanzaremos el asistente de Alta Disponibilidad pulsando en el panel derecho en "configurar un servicio"

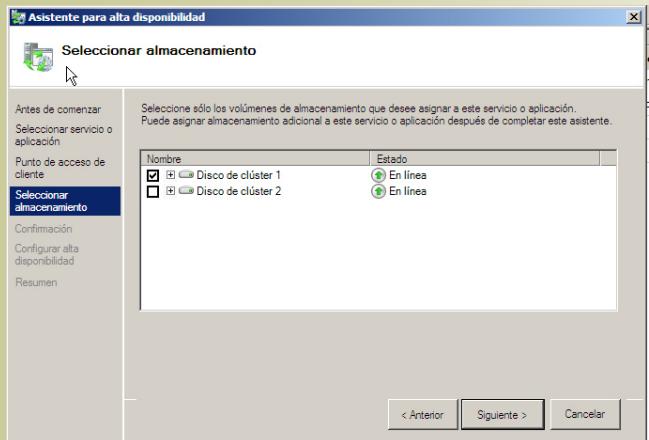


Elegiremos de los que ofrece "Servidor de archivos".

Nos pedirá le digamos la IP y nombre de acceso para los clientes. La IP pertenecerá a la red LAN, pero no será la misma que pusimos para administración para mayor seguridad. En nuestro caso usaremos 192.168.1.250/24

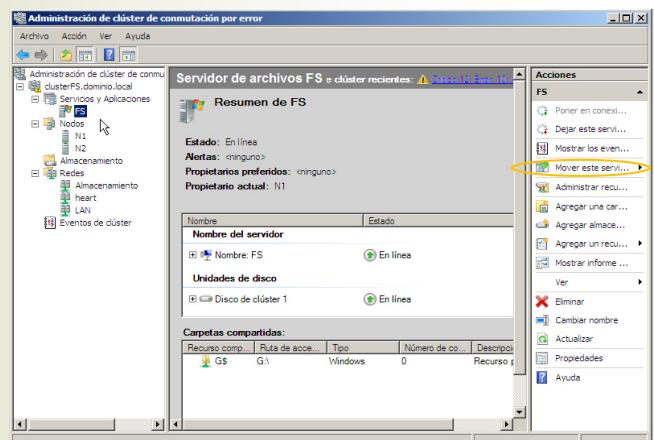


Y al seleccionar discos, sólo pincharemos uno de los dos que hemos asignado al cluster:

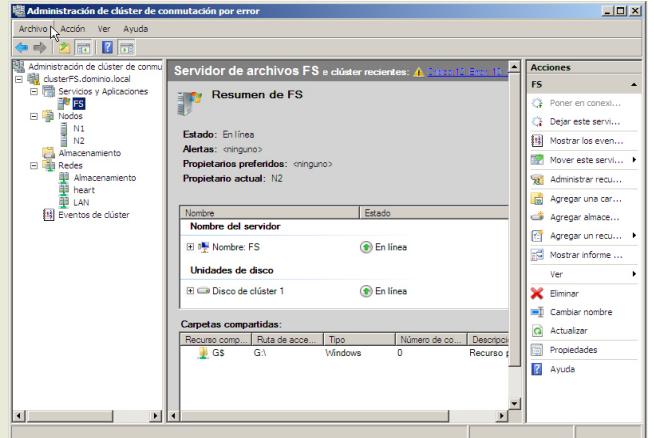
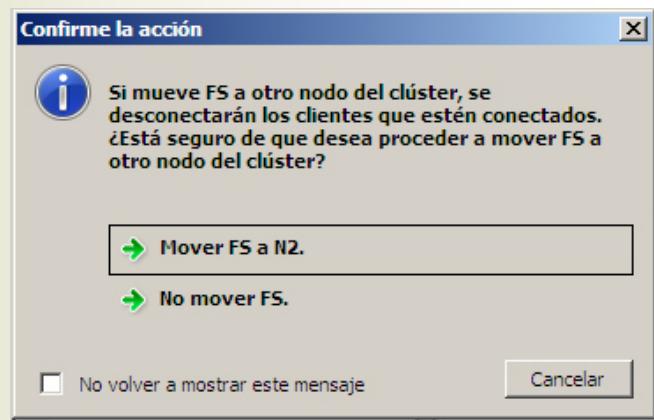


Ya tenemos nuestra primera aplicación o servicio colgada del clúster, ahora simplemente una comprobación. Como podemos observar en el panel informativo central, este servicio se está ejecutando en el nodo2 ("Propietario actual N2"). Tenemos que comprobar que se mueve correctamente entre los nodos.

Para ello, podemos forzarlo con botón derecho sobre el servicio/aplicación "Mover este servicio o aplicación a otro nodo" > "Mover al nodo N2",



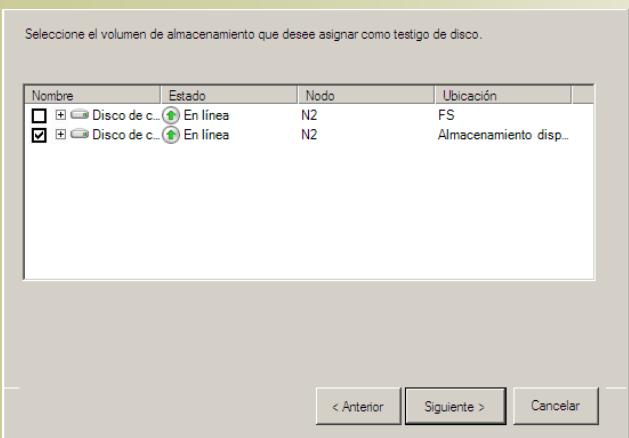
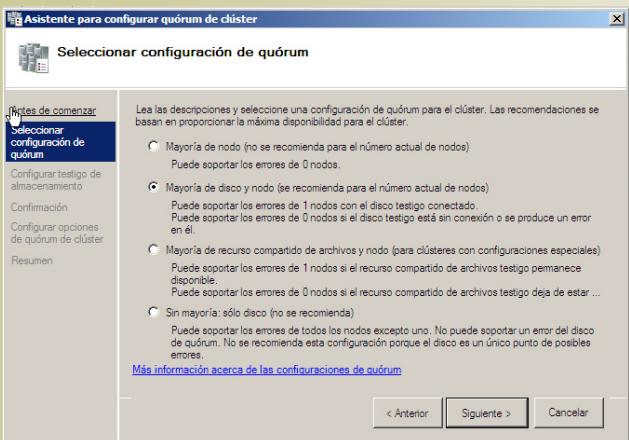
Confirmamos que queremos moverlo, ya que quien esté conectado podrá perder la conexión durante unos segundos



Listo, tras esto, comprobamos que el servicio/aplicación ya se ejecuta en el otro nodo. Ahora ejecutaremos la misma prueba devolviéndolo a otro nodo, simplemente para comprobar que todo es correcto.

También podríamos apagar el nodo que tenga el servicio en uso para simular una caída. Pero previamente, veremos si la opción de quorum que tenemos es adecuada a nuestro numero de nodos, y si no es así, modificaremos:

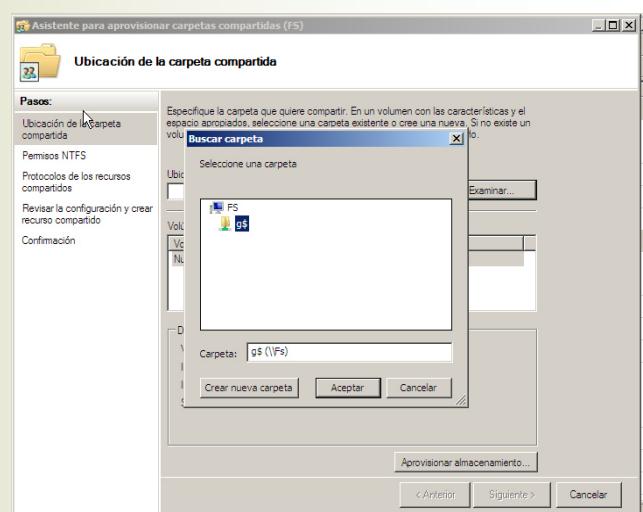
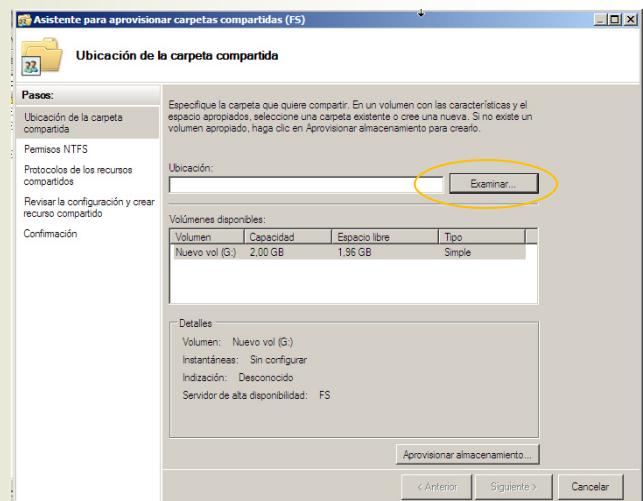
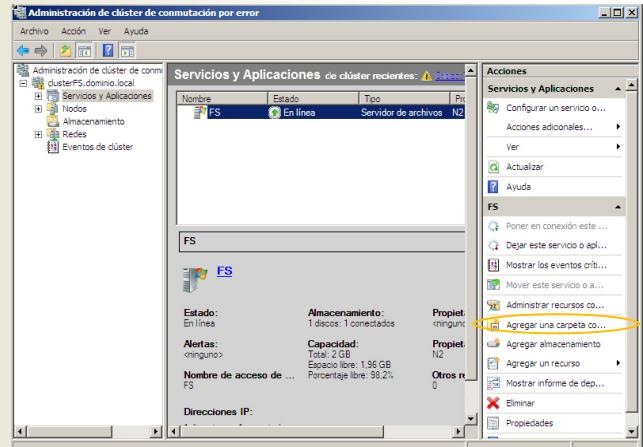
Acción/Acciones adicionales/configurar opciones quórum



Como podemos ver en la imagen, para nuestro caso, cluster de dos nodos, seleccionamos **mayoría de disco y nodo** y pinchando el disco de almacenamiento libre.

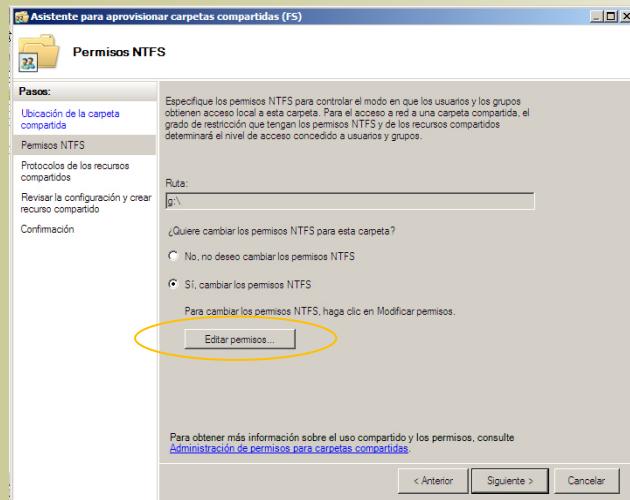
Para probar el servicio, necesitaremos crear una carpeta común en el cluster y agregar un equipo cliente al Active Directory para acceder como user1, el usuario de dominio que creamos inicialmente. Usaremos un XP.

Desde la administración del cluster, sobre el servicio creado, podemos ver en la parte derecha la opción de agregar una carpeta compartida.

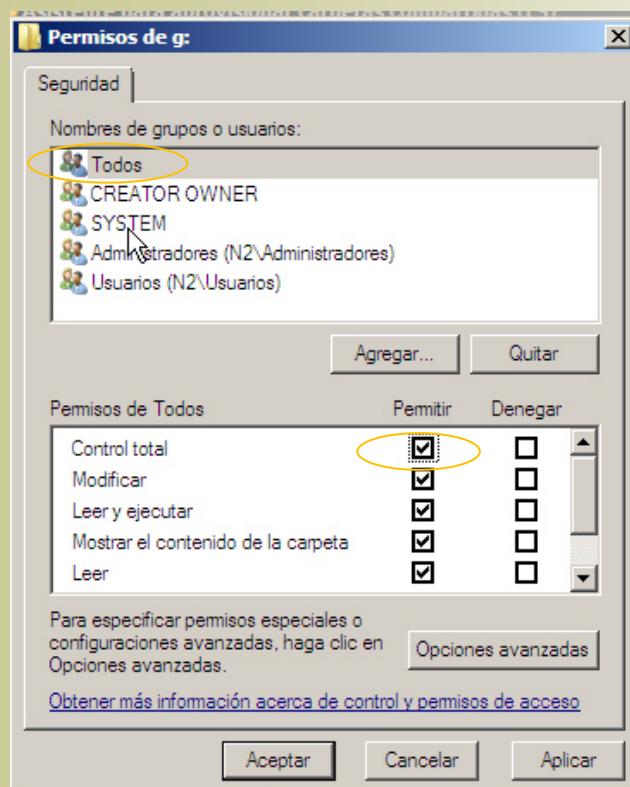


Elegimos la carpeta que nos aparece dentro del servicio por defecto.

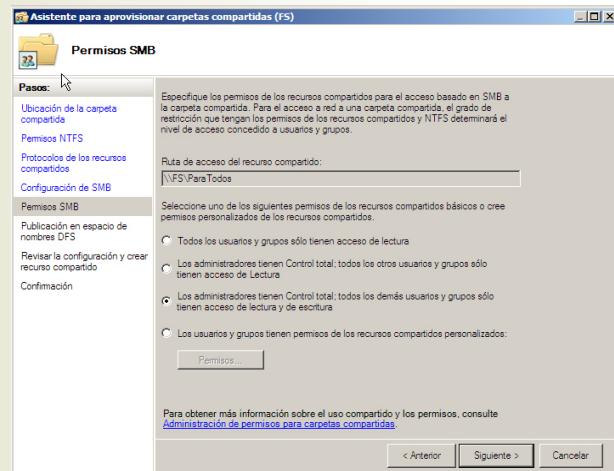
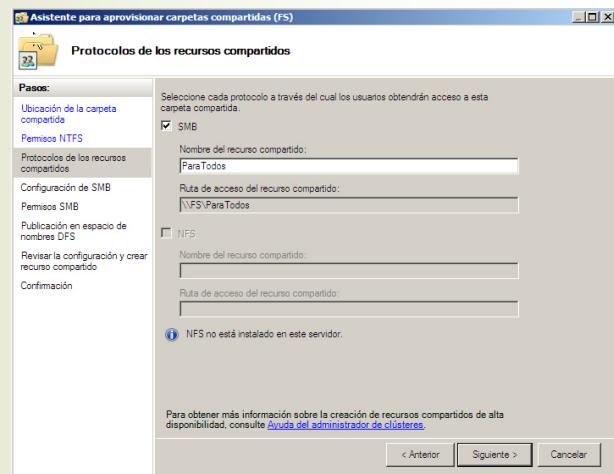
Pinchamos cambiar permisos ntfs para que nos permita editarlos:



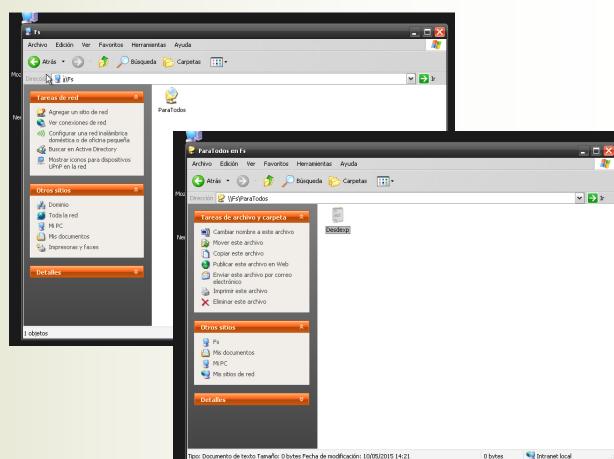
Agregamos "Todos" y damos los permisos que deseemos que tengan nuestros usuarios. Nosotros, para este ejemplo, daremos control total.



A continuación, nos pedirá seleccionemos protocolo para el recurso compartido. Elegiremos Samba.

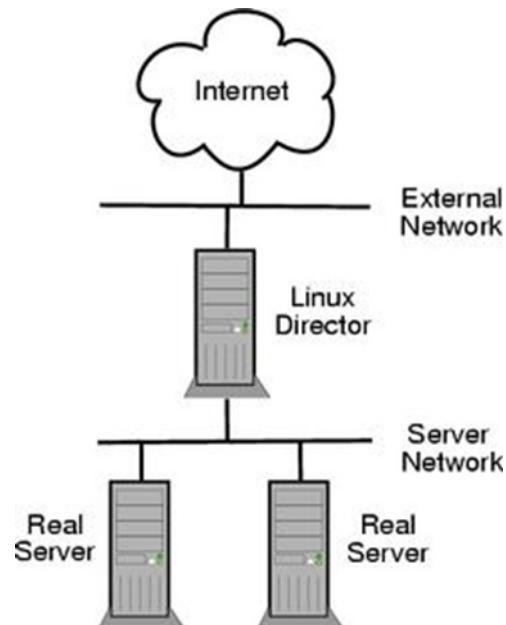


Probamos uniendo el XP al dominio y accediendo como user1 al servicio como unidad de red. Creamos una carpeta y cambiamos el nodo apagando. Podemos así ver, el correcto funcionamiento del cluster. Apenas perderemos conexión unos segundos y seguiremos teniendo acceso a la carpeta creada.



BALANCEO DE CARGA

Un clúster de balanceo de carga reparte las peticiones de servicio recibidas entre los diversos nodos servidores existentes. Este sistema tiene varias ventajas como la de ampliar la capacidad fácilmente tan sólo sumando nodos al clúster, cosa que hace que tengamos más capacidad para balancear y por tanto mayor capacidad de respuesta frente a peticiones. A su vez este sistema ofrece robustez puesto que ante la caída de uno de los nodos el servicio de balanceo de carga sigue funcionando mientras queden nodos disponibles.



Opciones que desechamos:

Openmosix. Es un sistema de clúster para Linux muy interesante, que consiste en un parche en el kernel responsable de migraciones transparentes de procesos, y herramientas de área de usuario para calibrar y administrar el clúster. Esto permite que no tengamos que reprogramar nuestras aplicaciones para que aprovechen el clúster. Los procesos no saben en qué nodo del clúster se ejecutan, y es el propio openMosix el responsable de "engaños", y redirigir las llamadas del sistema al nodo que actúa en ese momento de master. El parche para el kernel funciona en las versiones 2.4 y 2.6, aunque en esta última solo de forma experimental. Actualmente el desarrollo del proyecto está parado.

Aunque fue nuestra primera elección, tuvimos problemas al usar portátil, ya que requiere ratón estándar serie o PS/2 o USB IMPS/2 compatible. Además, es inseguro porque el kernel necesario está desactualizado.

Otra opción que parece más actualizada es LVS. Existe numeroso software que lo ha incorporado, algunos bastante conocidos como **Ultramonkey** y Piraña.

También es interesante el módulo de **webmin**, pero finalmente acabamos eligiendo Pirahna (para red hat).

Piranha cómo balanceador de cargas en centOS

Piranha es un paquete de software incluida en la familia Red Hat, esta compuesto por un servidor LVS (Linux Virtual Server) y un gestor del mismo, que permite administrar los servicios de la Web con un navegador a través de una interfaz gráfica.

LVS permite crear un clúster de balanceo de carga, en el cual hay un nodo que se encarga de gestionar y repartir las conexiones (nodo master LVS) entre todos los nodos esclavos del clúster. El servicio de datos debe residir en todos los nodos esclavos. LVS puede soportar sin problemas hasta 200 nodos esclavos.

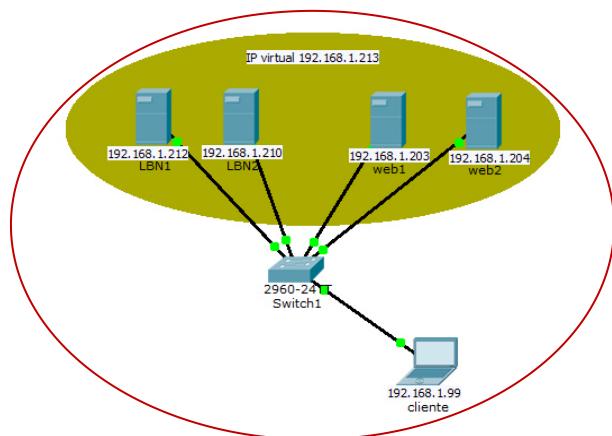
Piranha viene en los repositorios de los sistemas operativos basados en Red Hat. Por defecto escucha por el puerto 3636



PRÁCTICA BALANCEO DE CARGA RED HAT

Para la realización de esta práctica, hemos utilizado cuatro centOS. Dos que actúan como servidores webs, y dos para balancear la carga (Master y Backup), que distribuirán las peticiones web que realicen los clientes.

Los cuatro equipos conjuntamente, serán vistos como una única IP por el usuario, que compartirá la misma red.



En ambos centOS instalamos e iniciamos:

```
> yum install piranha ipvsadm
> service piranha-gui start
> chkconfig piranha-gui on
> chkconfig pulse on
```

Establecemos una contraseña para el usuario "piranha" que nos permitirá tener acceso a la interfaz gráfica:

```
> piranha-passwd
```

Para que el LVS transmita los paquetes de red adecuadamente a los servidores reales, es necesario activar el reenvío de paquetes en los nodos balanceadores editando:

```
> nano /etc/sysctl.conf
net.ipv4.ip_forward = 1
```

```
> sysctl -p
```

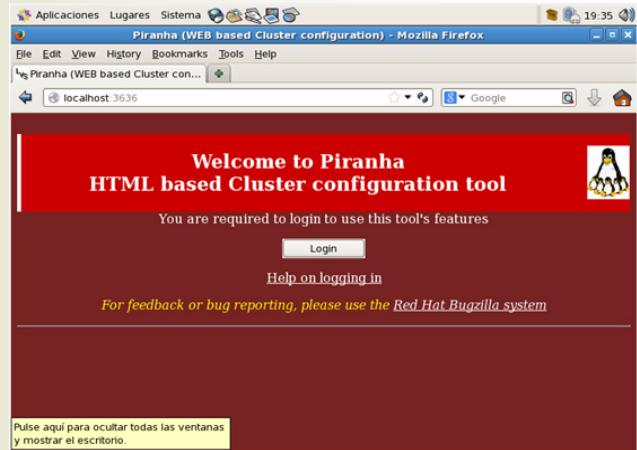
Antes de entrar al entorno gráfico de piranha activamos los servicios HTTP en los servidores reales

```
> service httpd start
```

Los servidores web, deben ofrecer la misma página al cliente, de tal forma que el proceso de balanceo sea transparente, bien por acceso a un "sitio" común, que puede ser compartido por ambos, o bien, si la página es estática puede copiarse en cada uno de los servers.

Nosotros, para poder realizar las comprobaciones, vamos a realizar lo contrario. Es decir, modificaremos el archivo index.html de cada server apache para identificarlos. Así, mediante peticiones consecutivas a la misma IP, debemos ver páginas diferentes, mostrando así que nos la sirven distintos equipos.

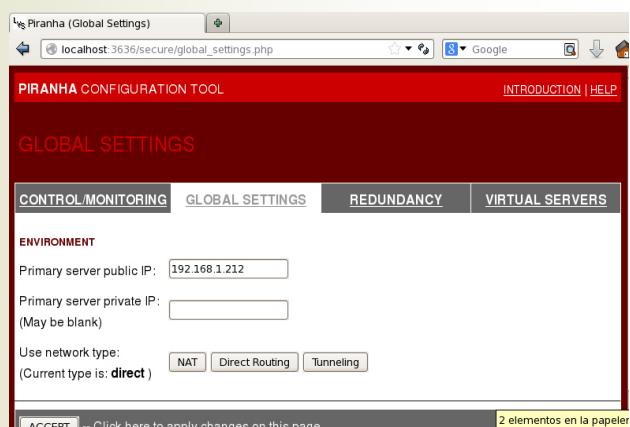
Accedemos a la interfaz de administración de Piraña, desde un navegador mediante la IP del nodo balanceador y el puerto 3636. El usuario es "piranha" y la contraseña la que se asignó anteriormente con el comando piranha-passwd.



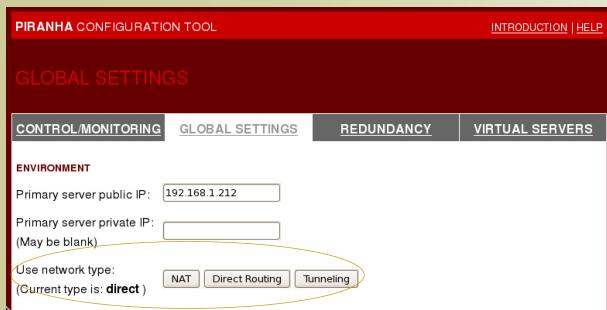
Esta interfaz, se encargará de modificar el fichero de configuración de lvs /etc/sysconfig/ha/lvs.cf sin posibles errores de sintaxis.

Ambos nodos deben tener idéntica configuración, por lo que otra manera de hacerse sería configurando con piranha-gui sólo el nodo master y copiando su fichero lvs.cf al backup con scp.

Una vez logeados debemos configurar la dirección IP primaria del servidor público, es decir, la IP del centOS "master", que enrutará balanceando, en la pestaña GLOBAL SETTINGS. Lo haremos en ambos nodos.

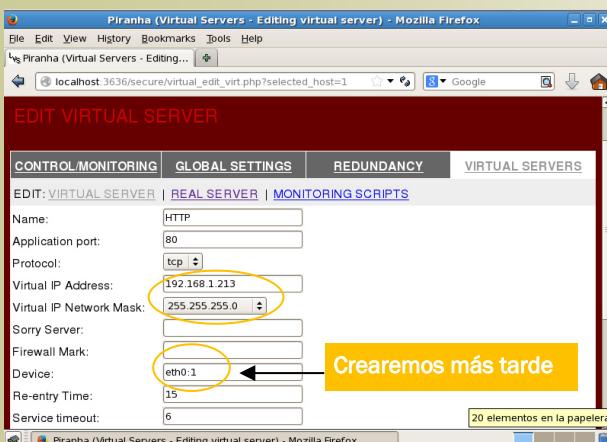


En la interfaz gráfica de piranha, en la pestaña GLOBAL SETTING, en su parte inferior, podemos localizar tres opciones como formas de balanceo:

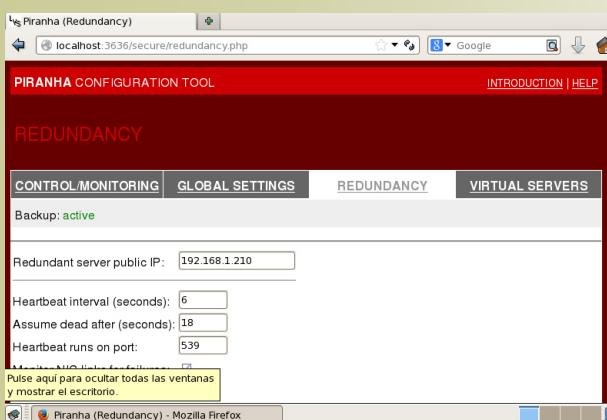


Nosotros escogeremos la más eficiente, Direct Routing, que ya viene como predeterminada.

También en ambos nodos, iremos a la pestaña VIRTUAL SERVERS para indicar la IP virtual que englobará el conjunto de los cuatro centOS. U otra manera de decirlo, la IP del servicio.



En la pestaña REDUNDANCY, indicaremos la IP de nuestro “backup”. También se realizará en ambos nodos. El resto de valores para comprobación del estado del estado de los nodos, los dejamos tal como están por defecto.



BALANCEADO

Distintos métodos

NAT (Network Address Translation). La máquina que recibe la información no es la destinataria. Recibe dicha información en forma de paquetes, y deberá reescribirlos sustituyendo su propia dirección con la de la máquina que realizó la petición (nos referimos a direcciones tanto físicas, MAC, como lógicas, IP). Una vez reescrito el paquete de la forma correcta el balanceador se encargará de enviar los paquetes por la interface adecuada para que le lleguen al destino verdadero.

Cuando el servidor responda lo hará al balanceador y este reescribirá el paquete, otra vez, poniendo en los paquetes la dirección del cliente que solicitó la información.

El problema es que esta sobreescritura de los paquetes trae consigo una carga de CPU, que puede llegar a ser un cuello de botella, y un mayor consumo del ancho de banda.

Utilizando **IP Tunneling** el balanceador únicamente tendrá que hacerse cargo de las peticiones de los clientes y enviarlas a los servidores, siendo estos mismos los que responderán a los clientes. De esta forma el balanceador de carga puede manejar mas nodos, es decir el servicio es mas escalable.

El balanceador recibe un paquete, lo encapsula en un datagrama IP y lo manda a uno de los servidores. Cuando el servidor lo recibe sólo tendrá que desencapsularlo y responderá directamente al cliente

Con **Direct Routing**, al igual que en IP Tunneling el balanceador sólo gestionará las peticiones del cliente hacia el servidor con lo cual es una solución altamente escalable.

La dirección virtual (VIP) es compartida por el balanceador y los servidores. De esta manera el balanceador recibe las peticiones y las envía a los servidores que procesan las peticiones y dan servicio directamente a los clientes.

En esta solución es necesario que una de las interfaces del balanceador y los servidores estén en el mismo segmento físico de red ya que el balanceador de carga cambiará su dirección física, MAC, en la trama por la dirección física de uno de los servidores que tendrá un alias con la dirección VIP.

El balanceador guarda una tabla de conexiones y cuando le llega un paquete determina si ya existe una conexión abierta y de ser así qué servidor real es el que está sirviéndola para enviarle el paquete.

Cuando utilizamos Tunneling o Direct Routing tanto el balanceador como los servidores comparten una dirección IP (VIP) y esto puede traer consigo problemas si los平衡adores y los servidores están en la misma red: El problema ARP, que analizaremos y solucionaremos más adelante.

Una vez definidos nuestros nodos master y backup, añadiremos la IP virtual y las IPs de nuestros apaches. Para ello, en la pestaña VIRTUAL SERVERS pinchamos en REAL SERVER.

Como ya vimos, al pinchar ADD nos aparece un formulario para determinar nuestra IP global del cluster, puerto y protocolo que utilizará.

CONTROL/MONITORING		GLOBAL SETTINGS		REDUNDANCY		VIRTUAL SERVERS	
STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE	
	up	HTTP	192.168.1.213	255.255.255.255	80	tcp	5

ADD DELETE EDIT (DE)ACTIVATE

Si volvemos a editar, podemos ver que disponemos de una pestaña denominada REAL SERVERS, donde añadiremos y editaremos nuestro servidores web:

STATUS	NAME	ADDRESS	
	up	web1	192.168.1.203
	up	web2	192.168.1.204

ADD DELETE EDIT (DE)ACTIVATE CANCEL

Podemos incluir con pirahna hasta 200 Real server.

En los Real servers añadiremos como otra IP de la misma NIC, la virtual del cluster. Para ello, añadimos a /etc/rc.local para que después de cada reinicio se ejecute

```
ifconfig eth0:1 192.168.1.213 netmask 255.255.255.0
broadcast 192.168.1.255 up
```

OJO!!

Si la interfaz eth0:1 no se crea automáticamente, podemos crear el archivo correspondiente en los servidores:
`/etc/sysconfig/network-scripts/ifcfg-eth0:1`

ARP

El problema

Una situación a resolver en esta configuración es el "Problema ARP".

Como varios equipos tienen la misma IP, si la publican ante un requerimiento, habrá una competencia por el paquete, llamado "Race Condition".

Cuando llega una petición de un cliente, esta llegará desde fuera de la red, con lo cual el router de esa red hará una petición ARP para obtener la MAC de la máquina con la IP.

En esa red hay varias máquinas con la misma IP virtual (los balanceadores y los servidores apache) con lo cual cualquiera de ellas podría, responder a la petición. Pero el paquete debe ir destinado al balanceador, no a los servidores apache.

Si uno de los servidores web respondiera a la petición ARP el router tendría en su tabla ARP la dirección física del servidor y todos los paquetes se los enviará directamente al servidor sin utilizar el balanceador.

Para evitar eso, debemos asegurarnos que el único que publica la dirección IP Virtual es el director, por lo que debemos deshabilitar la publicación ARP en los servidores reales.

Solución al problema de ARP mediante ARPTABLES: Instalamos arptables_jf en ambos apaches.

```
> yum install arptables_jf
```

Configuramos en web server 1 (REAL SERVER 1)

```
> arptables -A IN -d 198.168.1.213 -j DROP
```

```
> arptables -A OUT -d 198.168.1.213 -j mangle -mangle-ip-s 198.168.1.203
```

Configuramos en web server 2 (REAL SERVER 2)

```
> arptables -A IN -d 198.168.1.213 -j DROP
```

```
> arptables -A OUT -d 198.168.1.213 -j mangle -mangle-ip-s 198.168.1.204
```

Guardamos y añadimos para ejecutar al inicio en ambos:

```
> service arptables_jf save
```

```
> chkconfig arptables_jf on
```

Podemos testear que todo esté correcto mediante:

>service arptables_jf status

```
[root@localhost ~]# service arptables_jf status
Tabla: filter
Chain IN (policy ACCEPT)
target    source-ip      destination-ip      source-hw      destinat
ion-hw   hlen  op        hrd      pro          anywhere     anywhere
DROP      anywhere      192.168.1.213      anywhere     anywhere
any      any      any      any
Chain OUT (policy ACCEPT)
target    source-ip      destination-ip      source-hw      destinat
ion-hw   hlen  op        hrd      pro          anywhere     anywhere
mangle   anywhere      192.168.1.213      anywhere     anywhere
any      any      any      any      -mangle-ip-s 192.168.1.204
Chain FORWARD (policy ACCEPT)
target    source-ip      destination-ip      source-hw      destinat
ion-hw   hlen  op        hrd      pro          anywhere     anywhere
[root@localhost ~]#
```

Una segunda opción para solucionar el problema ARP es con IPTABLES, pero puede crear sobrecarga por el enmascaramiento y reenvío de cada paquete.

iptables -t nat -A PREROUTING -p tcp -d 192.168.1.213 – dport 80 -j REDIRECT

Comprobamos el cluster en los balanceadores mediante
> ipvsadm -L

```
root@localhost:~#
Archivo Editar Ver Terminal Solapas Ayuda
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP 192.168.1.213:http lblc
-> 192.168.1.204:http           Route 1    0      0
-> 192.168.1.203:http           Route 1    0      0
[root@localhost ~]#
```

Aquí vemos que esta la IP virtual 192.168.1.213 y depende los dos real server (webs) 192.168.1.203 y 192.168.1.204

También podemos utilizar

>watch ipvsadm

Con esto, ya tenemos configurado mínimamente el cluster, y ya podemos ver el resultado desde un equipo cliente.

Para ello, solicitaremos en el navegador el acceso a la IP virtual.



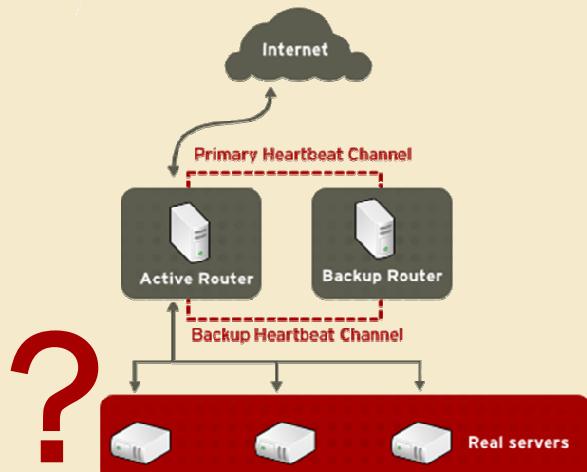
Como podemos apreciar, hemos recibido una respuesta desde nuestro server web2.

¿Porqué web2 y no web1? La contestación a esta pregunta hay que buscarla en la configuración del cluster. En concreto en el algoritmo de balanceo de carga que hayamos elegido. En nuestro caso, no habíamos entrado a configurar, por lo que tenemos la opción por defecto.

Probamos a cambiar y utilizar distintos tipos de algoritmos y pesos para contrastar el resultado.

ALGORITMOS

Algoritmos de balanceo de carga



Aunque existen múltiples algoritmos, los más utilizados son:

rr: Round Robin: distribuye la carga en forma equitativa entre los servidores, una petición a cada uno.

wrr: Weighted Round Robin: proporcional a su peso. A las máquinas con mayor capacidad de procesamiento se les dará un mayor peso (weight). Servidores con mas peso reciben mas carga.

lc: Least-Connection: Con este algoritmo las peticiones se enviarán al servidor que menos conexiones tiene sirviendo en ese momento.

wlc: Weighted Least-Connection: mas carga a servidores con menos conexiones ponderando su peso.

En ambos nodos: entramos en la pestaña VIRTUAL SERVER y editamos nuestra entrada de IP Virtual.

Dentro, podremos desplegar la opción "Device", donde encontramos todos los algoritmos disponibles en "Scheduling". Escogemos Round Robin.

Reiniciamos el servicio pulse y comprobamos:

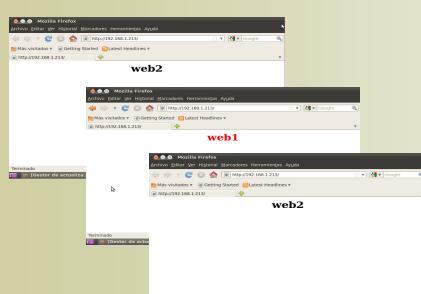
```
>service pulse restart
```

```
>ipvsadm -L
```

Veremos que ahora el algoritmo es rr (Round Robin)

```
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP 192.168.1.213:http rr
  -> 192.168.1.204:http          Route   1      0      3
  -> 192.168.1.203:http          Route   1      0      3
[root@localhost ~]#
```

Probamos desde nuestro cliente a solicitar la web de 192.168.1.213 reiterativamente, y observamos que la web es ofrecida por nuestros servidores apaches alternativamente.



Probamos ahora a repetir la operación, pero eligiendo Weighted Round Robin:

Pero para darle peso, necesitaremos entrar en el server real que queramos priorizar, y editar desde la pestaña REAL SERVER:

Aquí vemos como le hemos dado 10, pero podemos probar con distintos pesos, 3, 200, 4,

Reiniciamos pulse y comprobamos.

```
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP 192.168.1.213:http rr
  -> 192.168.1.204:http          Route   1      0      3
  -> 192.168.1.203:http          Route   1      0      3
[root@localhost ~]# service pulse restart
Shutting down pulse: [ OK ]
Starting pulse: [ OK ]
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP 192.168.1.213:http rr
  -> 192.168.1.204:http          Route   1      0      0
  -> 192.168.1.203:http          Route   4      0      0
[root@localhost ~]#
```

Probaremos ahora a realizar peticiones desde nuestro cliente.

Puede ocurrir, que tras cambiar, nuestro cluster tarde en responder. Si fuese el caso, podemos renovar la tabla ARP del cliente, y pedir ping a cada elemento del cluster para forzar el cambio y respuesta.

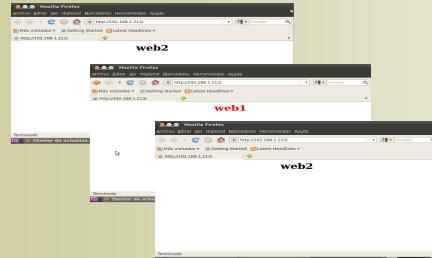
Por último, efectuaremos pruebas del funcionamiento de la redundancia de nuestros nodos, y de la ocultación de fallos y caídas de nuestros apaches.

Volvemos a configurar como Round Robin, por ser más útil para mostrar el balanceo.

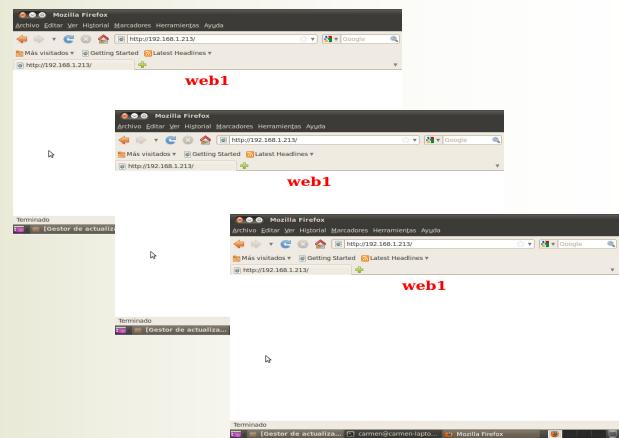
Apagamos el nodo1 que es nuestro master y comprobamos mediante el comando de estadísticas de administración, como nodo2 rápidamente se erige como balanceador:

```
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port Forward Weight ActiveConn InActConn
[root@localhost ~]# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.1.213:http rr
  -> 192.168.1.204:http      Route    2      0      0
  -> 192.168.1.203:http      Route    2      0      0
[root@localhost ~]#
```

Probamos desde nuestro cliente, con perfecto resultado:



Probamos ahora a simular la caída de un servidor web real. Apagamos web2 y volvemos a probar desde el cliente:



Obviamente, al haber configurado distintas web podemos apreciar la diferencia. Pero caso de haberse creado iguales, el cliente no notaría cambio alguno, constituyendo así un sistema de Alta Disponibilidad.

Finalmente, esta práctica se trasladó a equipos reales, pero con técnica de virtualización.



VIRTUALIZACIÓN DEL SERVIDOR

La virtualización del hardware implica utilizar software para crear máquinas virtuales (VM) que emulan un host físico. Esto crea un entorno de sistema operativo independiente que es, lógicamente, aislado del servidor host. Al ofrecer varias máquinas virtuales a la vez, este enfoque permite que varios sistemas operativos corran simultáneamente en una única máquina física.

En lugar de comprar varios servidores dedicados a funciones específicas que luego estarán subutilizados, la virtualización de servidores permite que las cargas de trabajo se consoliden en un número más reducido de servidores físicos plenamente utilizados.

Y es que en cuanto a aprovechamiento del hardware, un mal extendido en los CPDs actuales es el gran número de servidores, muchos de ellos infrautilizados. Si se virtualiza un número de esos sistemas en un solo equipo físico, se ahorrará energía, espacio, y capacidad de refrigeración, pero además tendremos otra serie de ventajas como son:

Aislamiento: las máquinas virtuales son totalmente independientes, entre sí y con el hypervisor. Por tanto un fallo en una aplicación o en una máquina virtual afectará únicamente a esa máquina virtual. El resto de máquinas virtuales y el hypervisor seguirán funcionando normalmente.

Seguridad: cada máquina tiene un acceso privilegiado (root o administrador) independiente. Por tanto, un ataque de seguridad en una máquina virtual sólo afectará a esa máquina.

Flexibilidad: podemos crear las máquinas virtuales con las características de CPU, memoria, disco y red que necesitemos, sin necesidad de “comprar” un ordenador con esas características. También podemos tener máquinas virtuales con distintos sistemas operativos, ejecutándose dentro de una misma máquina física.

Agilidad: la creación de una máquina virtual es un proceso muy rápido. Por tanto, si necesitamos un nuevo servidor lo podremos tener casi al instante, sin pasar por el proceso de compra, instalación, configuración, etc.

Portabilidad: toda la configuración de una máquina virtual reside en ficheros. Esto hace que sea muy fácil clonar o transportar la máquina de un servidor a otro, simplemente copiando y moviendo los ficheros que encapsulan la máquina virtual.

Recuperación en caso de fallo: si se dispone de una copia de la máquina virtual (ficheros o snapshots), en caso de desastre la recuperación será muy rápida. No es necesario reinstalar, recuperar backups y otros procedimientos largos que se aplican en las máquinas físicas.

Entorno seguro para pruebas y modificaciones.

Todas estas ventajas tienen un precio, que consiste fundamentalmente en una pérdida de rendimiento, es decir, una aplicación generalmente correrá más despacio en una máquina virtual que en un servidor físico.

ALTO RENDIMIENTO

El objetivo principal de un Cluster de Alto Rendimiento ó HPCC "High Performance Computing Cluster" es alcanzar el mayor rendimiento en la velocidad de proceso de datos para la computación paralela.

Este tipo de tecnología nos permite que un conjunto de computadoras trabajen en paralelo, dividiendo el trabajo en varias tareas más pequeñas las cuales se pueden desarrollar en el mismo tiempo, sin necesidad de esperar la finalización de otras.

Este tipo de cluster, ha sido fundamental en el desarrollo de importantes investigaciones, principalmente en las áreas de física, matemáticas e ingeniería.

Otros tipos de aplicaciones que aprovechan los clústeres de cálculo de manera usual son los modelos financieros (un algoritmo o una fórmula se ejecutan miles de veces cada una de ella con datos distintos), la animación informática (aplicación de efectos de textura e iluminación a cada fotograma de una película), y el descifrado de códigos.

Hace unos años, los sistemas de alto rendimiento (a los que se hacía referencia habitualmente como "supercomputación") estaban dominados por sistemas grandes y especializados (costosos) que se encontraban principalmente en centros de investigación. No obstante, a medida que la capacidad de cálculo de los sistemas pequeños ha aumentado, ha cambiado la

relación costo-rendimiento y las cargas de cálculo se han desplazado a los sistemas de PC .

Fue en 1994, cuando se integró el primer cluster de PCs en el Centro de Vuelos Espaciales Goddard de la NASA. Los investigadores de la NASA le dieron el nombre de Beowulf a este cluster, en honor del héroe de las leyendas medievales. Beowulf es una de las obras cumbre de la literatura anglosajona protagonizada por un héroe homónimo con la fuerza de unos treinta hombres en la palma de su mano, que derrotó al monstruo gigante Grendel.

Por tanto, cuando hablamos de Beowulf , hablamos de un sistema de cómputo paralelo basado en clusters de ordenadores personales conectados a través de redes informáticas standard, sin el uso de equipos desarrollados específicamente para la computación paralela. A ello se debe ser también conocido como la supercomputadora de los pobres.

Para ello, utiliza una estructura de un nodo maestro, que es el que permite el contacto con el exterior, y un indeterminado número de nodos esclavos, que trabajan paralelamente.



PRÁCTICA ALTO RENDIMIENTO CLUSTER BEOWULF

Para esta práctica utilizaremos un software español, mundialmente reconocido: ABC linux. Es una distribución basada en Knoppix, que viene completamente preparada para el uso del cluster. Incluye compiladores para programación paralela y herramientas de medición del rendimiento. Utiliza PXE para cargar en memoria en los nodos esclavos desde el nodo maestro.

Esta práctica se ha realizado previamente mediante virtualización, y luego se ha trasladado a equipos reales. En concreto hemos utilizado los centos usados en la anterior práctica para actuar de nodos esclavos. Como veremos a continuación, dichos nodos trabajarán sobre RAM, y no “pisarán” la instalación ya afectuada.

Para poder construir un cluster Beowulf con ABC GNU/Linux es preciso cumplir los siguientes requisitos:

Utilizar más de un PC, que todos los PCs integren 256 MB de memoria RAM y que al menos sean procesadores P3 500Mhz o equivalente. En caso de que se quiera construir un cluster en modo “live” no es preciso utilizar disco duro alguno.

Los PCs deben estar interconectados en la misma red.

En uno de los PCs introducimos CD de ABC GNU/Linux. Una vez arrancado nos dará a elegir:

```
ISOLINUX 3.63 Debian-2008-07-15 Copyright (C) 1994-2008 H. Peter Anvin
Automated Beowulf Cluster ABC GNU/Linux

IMPORTANT
user="master" password="master" hostname="master"

For the default live system, press ENTER or enter 'live'.

To start the installer directly, enter 'install'.

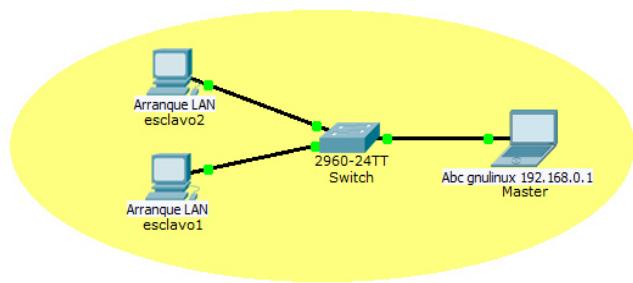
To verify the CD for errors, enter 'check'.

To run memtest86+, enter 'memtest'.

To boot from the first hard disk, enter 'hd'.

Iker Castaños Chavarri icastaños001@ikasle.ehu.es
Dept. Automatic Control and Systems Engineering
University of the Basque Country
http://www.ehu.es/AC

boot: -
```



Como puede ser observado en la captura de la pantalla, puede optarse por las siguientes opciones:

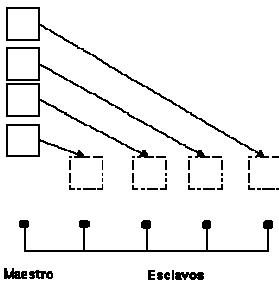
1. Modo “live”. Este modo de arranque se hace sin necesidad de instalación alguna. Todo el sistema arranca en RAM. Presionando la tecla “enter” o escribiendo “live” y presionando “enter”.
2. Modo “installer”. Mediante este modo se instalará la distribución en el disco duro del front-end. Es muy importante que durante la instalación se cree el usuario “master” con clave “master” y que al PC le pongamos como hostname “master”. Una vez instalado, su funcionamiento es idéntico al del modo “live”, solo que no se precisará disco alguno, el arranque será más rápido y los datos del “home” del usuario permanecerán aun apagando o reiniciando el sistema. Para optar por este modo se debe escribir “install” y presionar la tecla “enter”.
3. Modo “checkdisk”. Es utilizado para comprobar la integridad del soporte óptico. Toclear “check” y tecla “enter”.
4. Modo “memtest” Es utilizado para comprobar el estado de la memoria RAM. Para optar por este modo se debe teclear “memtest” y presionar “enter”.
5. En caso de que se opte por arrancar del disco duro tan solo se debe teclear “hd” y presionar la tecla “enter”.

EN PARALELO

Computación paralela

La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo)

Los programas informáticos paralelos son más difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de software, siendo las condiciones de carrera los más comunes. La comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.



MPI

Interfaz de Paso de Mensaje

MPI ("Message Passing Interface", Interfaz de Paso de Mensajes) es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que explotan la existencia de múltiples procesadores.

El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua, de manera similar a como se hace con los semáforos, monitores, etc.

Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación de sistemas distribuidos.

Tras el arranque, que nuestro caso hacemos en modo installer, accederemos al escritorio GNOME como se muestra en la siguiente captura de pantalla:



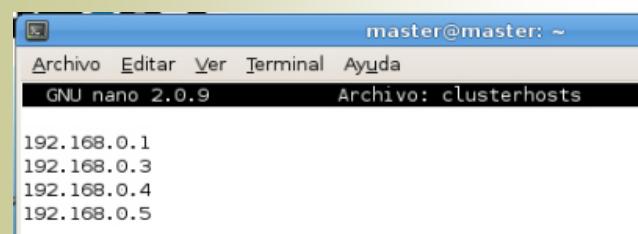
Una vez arrancado el front-end ya podemos arrancar los nodos esclavos.

Para ello, se debe configurar la BIOS especificando que el dispositivo de arranque sea la NIC mediante PXE (LAN). Una vez configurada la BIOS el arranque de cada nodo se llevará a cabo Y nos solicitará login:

```
* Starting system message bus dbus [ OK ]
* Starting Avahi mDNS/DNS-SD Daemon avahi-daemon [ OK ]
* Starting OpenBSD Secure Shell server sshd [ OK ]
* Starting portmap daemon...
* Already running.
* Starting Common Unix Printing System: cupsd [ OK ]
Starting Ganglia Monitor Daemon: gmond.
Starting Ganglia Monitor Meta-Daemon: gmetad.
* Starting NFS common utilities [ OK ]
* Exporting directories for NFS kernel daemon... [ OK ]
* Starting NFS kernel daemon [ OK ]
* Starting powernouveau...
* CPU frequency scaling not supported... [ OK ]
Not starting slurm-linnl
slurm.conf was not found in /etc/slurm-linnl
Please follow the instructions in /usr/share/doc/slurm-linnl/README.Debian
* Starting Hardware abstraction layer hal [ OK ]
* Starting System Tools Backend: system-tools-backends [ OK ]
landscape-client is not configured, please run landscape-config.
* Starting anac(h)ronistic cron anacron [ OK ]
* Starting deferred execution scheduler atd [ OK ]
* Starting periodic command scheduler crond [ OK ]
Automated Beowulf Cluster node tty1
node login: _
```

Tal como dijimos, master, master (no es necesario para su funcionamiento).

Todos los PCs del cluster han registrado sus IPs en el fichero "clusterhosts" que se encuentra en el "home" del usuario.



Si borramos el fichero para nueva instalación, hay que dejar al menos la IP del maestro para poder lanzar el demonio lam.

En nuestro caso, aparece el master (192.168.0.1) y los tres nodos que hemos arrancado.

Es preciso arrancar el daemon lam para poder ejecutar aplicaciones en paralelo sin ser root. El arranque del daemon lam lo llevaremos a cabo en una consola mediante el comando “lamboot -v clusterhosts” o clickeando el acceso “autolamboot”, el cual se encuentra en el escritorio del front-end.

```
master@master:~$ lamboot -v clusterhosts
LAM 7.1.2/MPI 2 C++/ROMIO - Indiana University
n-1<4209> ssi:boot:base:linear: booting n0 (192.168.0.1)
n-1<4209> ssi:boot:base:linear: booting n1 (192.168.0.3)
n-1<4209> ssi:boot:base:linear: booting n2 (192.168.0.4)
n-1<4209> ssi:boot:base:linear: booting n3 (192.168.0.5)
n-1<4209> ssi:boot:base:linear: finished
master@master:~$
```

Una vez arrancado el daemon lam es preciso compilar la aplicación que vayamos a correr en paralelo mediante el comando “mpicc.openmpi codigofuente -o binario”.

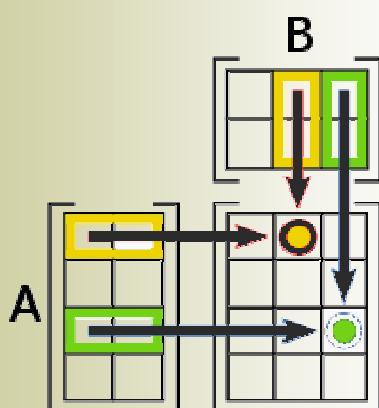
Y para correr la aplicación en paralelo es necesario ejecutar el comando “mpiexec.openmpi -n N binario”, siendo N el numero de nodos que quieran ser utilizados.

Efectuamos una pequeña muestra con el ejemplo del programa prueba.c

```
>mpicc.openmpi -o prueba.c prueba
>mpiexec.openmpi -c 3 prueba
```

```
master@master:~$ mpiexec.openmpi -n 1 prueba
hello from Node 0
master@master:~$ mpiexec.openmpi -n 3 prueba
hello from Node 0
hello from Node 1
hello from Node 2
master@master:~$
```

Ya estamos en disposición de probar con un programa paralelizable de multiplicación de matrices “matriz.c”



PRUEBA.C

Pequeño programa en C para comprobación de paso de mensaje

```
#include "stdio.h"
#include "mpi.h"
main (int argc, char **argv)
{
    int node;
    MPI_Init (&argc, &argv);
    MPI_Comm_rank (MPI_COMM_WORLD, &node);
    printf ("hello from Node %d\n", node);
    MPI_Finalize();
}
```

Como podemos observar, la multiplicación se puede trocear en pequeñas operaciones independientes.

En nuestro programa, definimos las matrices:

$$\begin{array}{cccc|c}
 & 1 & 4 & 7 & 3 & 1 & 43 \\
 & 2 & -5 & 8 & -1 & -4 & 75 \\
 & 3 & 6 & 9 & 5 & 7 & 57 \\
 & 4 & 4 & 2 & 0 & 3 & 2
 \end{array} \times = \begin{array}{c}
 \end{array}$$

4x4 4x1 4x1

Cuyo resultado matemático completo y secuencial será la matriz A_Exact y el resultado resultante de trocear y asignar tareas independientes a un determinado número de nodos será A.

MATRIZ.C

Programa de multiplicación de dos matrices, que nos permitirá comprobar el rendimiento del cluster.

```
#include "stdio.h"
#include "mpi.h"
#define NCOLS 4

int main(int argc, char **argv) {
    int i,j,k,l;
    int ierr, rank, size, root;
    float A[NCOLS];
    float Apart[NCOLS];
    float Bpart[NCOLS];
    float C[NCOLS];
    float A_exact[NCOLS];
    float B[NCOLS][NCOLS];
    float Cpart[1];
    root = 0;
    /* Initiate MPI.*/
    ierr=MPI_Init(&argc, &argv);
    ierr=MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    ierr=MPI_Comm_size(MPI_COMM_WORLD, &size);
    /* Initialize B and C.*/
    if (rank == root) {
        B[0][0] = 1;
        B[0][1] = 2;
        B[0][2] = 3;
        B[0][3] = 4;
        B[1][0] = 4;
        B[1][1] = -5;
        B[1][2] = 6;
        B[1][3] = 4;
        B[2][0] = 7;
        B[2][1] = 8;
        B[2][2] = 9;
        B[2][3] = 2;
        B[3][0] = 3;
        B[3][1] = -1;
        B[3][2] = 5;
        B[3][3] = 0;
        C[0] = 1;
        C[1] = -4;
        C[2] = 7;
        C[3] = 3;
    }
    /* Put up a barrier until I/O is complete */
    ierr=MPI_BARRIER(MPI_COMM_WORLD);
    /* Scatter matrix B by rows.*/
    ierr=MPI_Scatter
        (B,NCOLS,MPI_FLOAT,Bpart,NCOLS,MPI_FLOAT,root,MPI_COMM_WORLD);
    /* Scatter matrix C by columns.*/
    ierr=MPI_Scatter
        (C,1,MPI_FLOAT,Cpart,1,MPI_FLOAT,root,MPI_COMM_WORLD);
    /* Do the vector-scalar multiplication.*/
    for(j=0;j<NCOLS;j++) Apart[j] = Cpart[0]*Bpart[j];
    /* Reduce to matrix A.*/
    ierr=MPI_Reduce
        (Apart,A,NCOLS,MPI_FLOAT,MPI_SUM,root,MPI_COMM_WORLD);
    if (rank == 0) {printf("\nThis is the result of the parallel
computation:\n\n");
        printf("A[0]=%g\n",A[0]);
        printf("A[1]=%g\n",A[1]);
        printf("A[2]=%g\n",A[2]);
        printf("A[3]=%g\n",A[3]);
        for(k=0;k<NCOLS;k++) {
            A_exact[k] = 0.0;
            for(l=0;l<NCOLS;l++) {
                A_exact[k] += C[l]*B[l][k];
            }
        }
        printf("\nThis is the result of the serial computation:\n\n");
        printf("A_exact[0]=%g\n",A_exact[0]);
        printf("A_exact[1]=%g\n",A_exact[1]);
        printf("A_exact[2]=%g\n",A_exact[2]);
        printf("A_exact[3]=%g\n",A_exact[3]);
    }
    MPI_Finalize();
}
```

```
master@master:~$ mpiexec.openmpi -n 1 matriz
This is the result of the parallel computation:
A[0]=1
A[1]=2
A[2]=3
A[3]=4
This is the result of the serial computation:
A_exact[0]=43
A_exact[1]=75
A_exact[2]=57
A_exact[3]=2
master@master:~$
```

$$\begin{array}{|c|c|c|c|} \hline 1 & 4 & 7 & 3 \\ \hline 2 & -5 & 8 & -1 \\ \hline 3 & 6 & 9 & 5 \\ \hline 4 & 4 & 2 & 0 \\ \hline \end{array} \times \begin{array}{|c|} \hline 1 \\ \hline -4 \\ \hline 7 \\ \hline 3 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline 4 \\ \hline \end{array}$$

4x1 1x1 4x1

```
master@master:~$ mpiexec.openmpi -n 2 matriz
This is the result of the parallel computation:
A[0]=-15
A[1]=22
A[2]=-21
A[3]=-12
This is the result of the serial computation:
A_exact[0]=43
A_exact[1]=75
A_exact[2]=57
A_exact[3]=2
```

$$\begin{array}{|c|c|} \hline 1 & 4 \\ \hline 2 & -5 \\ \hline 3 & 6 \\ \hline 4 & 4 \\ \hline \end{array} \times \begin{array}{|c|c|} \hline 7 & 3 \\ \hline 8 & -1 \\ \hline 9 & 5 \\ \hline 2 & 0 \\ \hline \end{array} \times \begin{array}{|c|} \hline 1 \\ \hline -4 \\ \hline 7 \\ \hline 3 \\ \hline \end{array} = \begin{array}{|c|} \hline -15 \\ \hline 22 \\ \hline -21 \\ \hline -12 \\ \hline \end{array}$$

4x2 2x1 4x1

```
master@master:~$ mpiexec.openmpi -n 3 matriz
This is the result of the parallel computation:
A[0]=34
A[1]=78
A[2]=42
A[3]=2
This is the result of the serial computation:
A_exact[0]=43
A_exact[1]=75
A_exact[2]=57
A_exact[3]=2
```

$$\begin{array}{|c|c|c|} \hline 1 & 4 & 7 \\ \hline 2 & -5 & 8 \\ \hline 3 & 6 & 9 \\ \hline 4 & 4 & 2 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 3 & & \\ \hline -1 & & \\ \hline 5 & & \\ \hline 0 & & \\ \hline \end{array} \times \begin{array}{|c|} \hline 1 \\ \hline -4 \\ \hline 7 \\ \hline 3 \\ \hline \end{array} = \begin{array}{|c|} \hline 34 \\ \hline 78 \\ \hline 42 \\ \hline 2 \\ \hline \end{array}$$

4x3 3x1 4x1

```
master@master:~$ mpiexec.openmpi -n 4 matriz
This is the result of the parallel computation:
A[0]=43
A[1]=75
A[2]=57
A[3]=2
This is the result of the serial computation:
A_exact[0]=43
A_exact[1]=75
A_exact[2]=57
A_exact[3]=2
master@master:~$
```

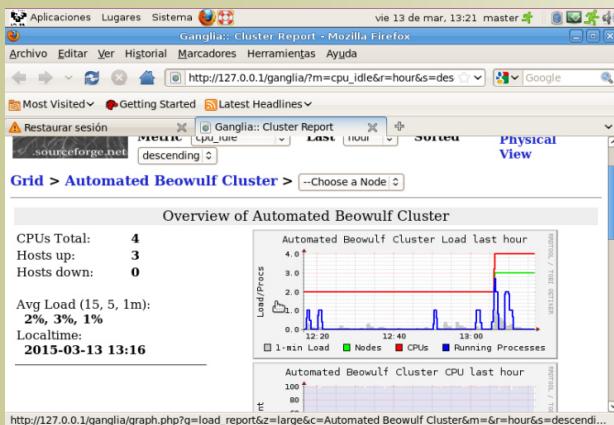
$$\begin{array}{|c|c|c|c|} \hline 1 & 4 & 7 & 3 \\ \hline 2 & -5 & 8 & -1 \\ \hline 3 & 6 & 9 & 5 \\ \hline 4 & 4 & 2 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|} \hline 1 \\ \hline -4 \\ \hline 7 \\ \hline 3 \\ \hline \end{array} = \begin{array}{|c|} \hline 43 \\ \hline 75 \\ \hline 57 \\ \hline 2 \\ \hline \end{array}$$

4x4 4x1 4x1

Podemos ver distintos resultados según el numero de nodos que utilicemos: Obviamente, coinciden al repartir la totalidad de las tareas (4) entre el mismo número de nodos.

Para la medición del rendimiento se utilizan distintas herramientas.

La distribución incluye un monitor de recursos llamado ganglia. Es posible acceder a el presionando el acceso directo llamado "Cluster monitor". A través del navegador Firefox se visualizara todos los recursos de CPU; memoria, red, etcétera.



Este software Ganglia, específico para clusters, nos ofrece una visión general del clustter, con el número de hosts y CPUS, y datos de uso de la CPU y la memoria RAM entre otros, global y de cada nodo de forma individualizada.

Otra herramienta para medir el rendimiento es el denominado Test de Linpack de rendimiento general.

El principio básico del test consiste en medir el tiempo empleado en la ejecución de ciertas instrucciones. En su mayoría son llamadas a BLAS (Basic Linear Algebra Subprograms), que realiza operaciones matriciales y vectoriales.

Existe una página en Internet que informa el "Top 500" de las computadoras basándose en el Linpack

<Http://www.top500.org>



Una vez vista la imagen de este supercomputador, se nos viene a la mente la conocida frase de que las comparaciones son odiosas, pero aun así, ha sido un orgullo y placer trabajar en la elaboración de este cluster de cuatro nodos esclavos:



RANK SITE

- | | |
|---|---|
| 1 | National Super Computer
Center in Guangzhou
China |
|---|---|

CLUSTERS

Junio 2015

CarmenJara

mcarmenjara@gmail.com

Este trabajo habría sido imposible sin mis profesores y profesoras.

Gracias de corazón a todos por compartir conocimiento y enseñarme a seguir aprendiendo.

INFORMACIÓN UTILIZADA

<http://www.recercat.cat/bitstream/handle/2072/219063/GutierrezSanmiguelAlfred-ETISa2011-12.pdf?sequence=1>

<https://albertomolina.wordpress.com/2012/03/04/sencillo-cluster-de-alta-disponibilidad-con-pacemaker-y-corosync/>

<http://clusterlabs.org/>

<http://www.bujarra.com/creando-un-cluster-de-alta-disponibilidad-en-microsoft-windows-server-2008/>

<http://www.sysprobs.com/configure-iscsi-server-pc-freenas-072-virtualbox-324>

http://www.guillesql.es/Articulos/Configurar_Microsoft_iSCSI_Target.aspx

<https://technet.microsoft.com/es-es/library/cc731844%28v=ws.10%29.aspx>

<http://informatica.gonzalonazareno.org/proyectos/2011-12/jmmr.pdf>

<http://www.ultramonkey.org/2.0.1/topologies/lb-overview.html>

<http://www.linuxvirtualserver.org/software/index.html>

<http://easylinuxtutorials.blogspot.com.es/2012/07/installing-configuring-linux-load.html>

<http://estefani.mx/2014/balanceo-de-cargas-rhel-centos.html>

<https://kezhong.wordpress.com/2010/03/28/setup-linux-loadbalancer-with-piranha-and-lvs-on-centos-5-4/>

<http://www.microsoft.com/spain/virtualizacion/products/server/default.mspx>

<http://www.techweek.es/virtualizacion/tech-labs/1003109005901/ventajas-desventajas-virtualizacion.1.html>

<http://blog.secaserver.com/2012/07/centos-configure-piranha-load-balancer-direct-routing-method/>

https://access.redhat.com/documentation/es-ES/Red_Hat_Enterprise_Linux/6/html-single/Virtual_Server_Administration/index.html

<http://es.slideshare.net/antoniobp/proyecto-login-plataforma-de-clustering-beowulf>

<http://abcgncnlinux.webnode.es/>

<http://acarus.uson.mx/choya.htm>